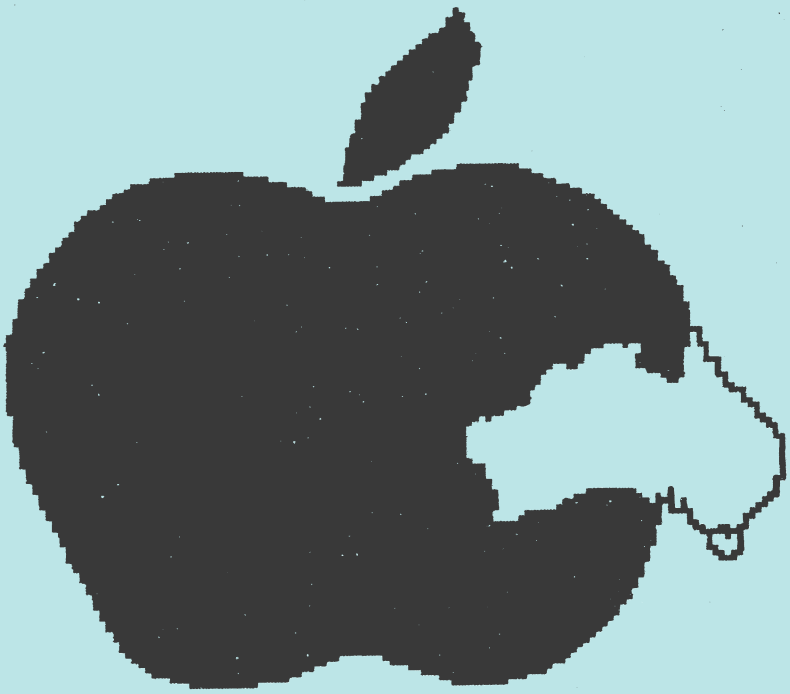
APPLECATIONS



apple users group

SYDNEY

AUSTRALIA

EXECUTIVE :
------PRESIDENT- Bruce Kehlet
VICE PRES- Michael McGuinness (047) 51-3092
SECRETARY- Colin Rutherford (02) 520-0926
TREASURER- Peter Kazacos (02) 705-6490
COMMITTEE : Graham Saint
George Tamindjis

CLUB MEETINGS : 6.30 p.m.-2nd. MONDAY of each month at the SYDNEY GRAMMAR SCHOOL, SCIENCE AUDITORIUM.
MEMBERSHIP : $15 Joining Fee, $15 Annual Subscription.

POSTAL ADDRESS : P.O.Box 505 BANKSTOWN, N.S.W., 2200
HAND DELIVERY : Computerland , 31 Market st. Sydney.

APPLECATIONS : Original material may be reproduced without approval. Please credit Author and source.
EDITOR : Hans Hoffman (02) 523-1412
10/33 The Esplanade, CRONULLA, N.S.W., 2230.
ADVERTISING : Bruce Stanley (02) 498-6972

LIBRARIAN : Don Riley (02) 451-2475

TRADING & B/P : Ed Mehrtens (02) 523-5320 a.h/543-3697 bus

PASCAL GROUP : Nancy Corbett (02) 919-5673

JULY * 1982

HIGHLIGHTS

DISK II PATENT

LADDER GAME

SCREEN SPLIT-2

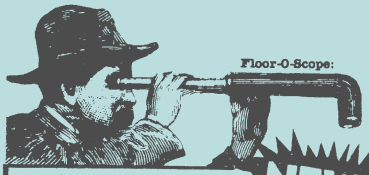
SLOT SEARCH

SOFTSTORY

REGISTERED BY AUSTRALIA POST - PUBLICATION No. NBH 3716

IMAGINEERING

"your software professionals"



NEW!

Apple Mechanic

Shape Writer/Byte-Zap Utility
by Bert Kersey

Another hot multiple-utility disk with nine useful, Listable, copyable and customizable programs—

Shape Writer: Put professional hi-res animation in your programs! From the keyboard, draw anything, then let your Apple write the shape table and store it on disk. Design custom typefaces too, any size, and special characters. LIST-able demo programs show **how to use shape tables** for animating your own games, graphic displays and **animated charts & graphs.** If you are into graphics, this valuable utility/learning tool is for you.

Byte Zap: A "must" utility. Rewrite any byte on a disk by loading a sector onto the screen for inspection. **Hex, decimal or ascii displays** optional. Examine bytes via cursor control; enter hex, decimal or ascii to change. Useful for creating illegal filenames, restoring deleted files, changing greeting program names, repairing or protecting disks, making DOS changes, and examining program files. Clear instructions show how Apple disk data is stored and how to access it.

Gosub City: A useful collection of sounds, text and hi-res tricks **for use in your programs.**

More too: Demo-writing programs, more hi-res utilities, excellent educational documentation... we'd like to go on, but this ad space is **expensive!**

☐ **Apple Mechanic** disk (48K min.)
☐ Beagle Bros **Apple Tip Book #5**
☐ **Peeks, Pokes & Pointers** Chart

Tip Disk #1

by Bert Kersey

100 programs from Beagle Bros' Tip Books 1, 2, 3 and 4—Hi-res, Lo-res and Text demos, and dozens of trick programs to make your Apple do things it's never done before! All 100 programs are listable, copyable and changeable, and each teaches another fact about making your Apple do its thing.

☐ **Tip Disk #1** on disk (32K or 48K)
☐ **Peeks, Pokes & Pointers** Chart

Note: No Tip Book with Tip Disk

DOS BOSS

DISK COMMAND EDITOR
by Bert Kersey & Jack Cassidy

A classic Apple utility you will ENJOY! Rename commands: "Catalog" can be "Cat", and so on. **Save-protect your programs;** An unauthorized copy attempt produces a "Not Copyable" message. **List-protection** too and one-key program selection from catalog. Catalog customizer: **Change Disk Volume message** to your title; Omit or alter file codes. Rewrite error messages: "Syntax Error" can be "Oops!!" or anything you want! Fascinating documentation included; Hours of good reading!

Any or all of Dos Boss's change features may be appended to your programs, so that anyone using your disks (booted or not) on any Apple will be formatting DOS the way you designed it.

☐ **Dos Boss** on Disk (32K or 48K)
☐ Beagle Bros **Apple Tip Book #2**
☐ **Peeks, Pokes & Pointers** Chart

Utility City

21 Useful Utilities on One Disk
by Bert Kersey

21 versatile utilities you can list, customize and back-up: **List Formatter** makes properly spaced and indented listings with printer page breaks; each statement on a new line with if-then's and loops called out; a great de-bugger! **Catalog in multiple-columns** and any page-width to printer or screen. Automatically post Run-number and last-used Date in your programs. Put **invisible functioning commands** in your listings. Access program lines in memory for garbage repair and "illegal" alteration. Quickly alphabetize and store info on disk. Run any program while another stays intact. Renumber to 65535. Save inverse, trick and **invisible file names.** Convert decimal to hex & binary, or INT to FP. Append programs. Dump text screen to printer... More too; **21 Programs Total!**

(All disks DOS 3.3 unless 3.2 requested)

ALL 21 PROGRAMS

☐ **Utility City** on disk (48K minimum)
☐ Beagle Bros **Apple Tip Book #3**
☐ **Peeks, Pokes & Pointers** Chart

More tips from the authors of Beneath Apple Dos . . .

BAG OF TRICKS

Bag of tricks is useful to beginners and experienced programmers alike. It includes many "hand holding" tutorials that assist you in repairing damaged diskettes and allow you to change sector ordering and reconstruct blown catalogs.

Apple — Cillin II™

Apple — Cillin II™ will verify over two dozen hardware operations, and either identify a specific problem or give your hardware a clean bill of health. Quickly and easily. Now with these new features:

- Linked Tests
- 6502 CPU Test
- Printed Results
- Disk Speed Test
- Boots from Any Slot

GOTO your Apple Dealer.

Most Apple Dealers carry our software. If your dealer doesn't, get on his case. He can have our disks in his store for you **within 2-3 days.** Or, if you want, you can order directly from us. Call our Toll Free Number (below) or drop us a note with a check or your Visa/MasterCard number & expiration date.

Bonuses With Every Disk!

Poke your Apple all night long with this free handy 11" x 17" reference poster! The most useable **PEEKS, POKES, POINTERS** and **CALLS,** scrounged up from every source imaginable!

Apple Tip Books too—

Each disk comes with a gold mine of valuable Apple information and hours of entertaining reading matter, including dozens of tips and keyboard experiments on all subjects— DOS, Copy Protection, Graphics, Shape Tables, Hardware and more. Sample programs too such as "Programming the Reset Key" and "Copy Stoppers". Each disk comes with its own unique book.

Alpha Plot

Hi-Res Graphics/Text Utility
by Bert Kersey & Jack Cassidy

Here are just a few of Alpha Plot's easy-to-use features. Compare with others on the market—

Hi-Res Drawing: Create hi-res pictures and charts with text on both pages of memory; all **appendable to your programs.** Optional Xdraw cursor (see lines before you draw). **Color mixes** and Reverse (opposite of background). Circles, Boxes, and Ellipses, filled or outlined. Scruncher **stores hi-res in 1/3 disk space.** Shifter redraws any portion of your picture on either hi-res screen. Also **superimpose images** and convert hi-res to lo-res and back for fascinating abstracts!

Hi-Res Text: Beautiful **upper & lower case** with descenders (no hardware required). Color or reverse characters positionable anywhere (no vtab/htab restriction). Professional-looking **proportional spacing** with adjustable height, leading (line spacing), and kerning (letter spacing). Multi-directional typing too for charts!

☐ **Alpha Plot** on Disk (48K minimum)
☐ Beagle Bros **Apple Tip Book #4**
☐ **Peeks, Pokes & Pointers** Chart

ON RELEASE NOW

at your local Apple dealer and our dealer of the month -

PETER SANDYS CITY PERSONAL COMPUTER (02) 233 8992

or IMAGINEERING (02) 358 3011

* C O N T E N T S * JULY 1982 *

* APPLECATIONS IS EDITED
USING SANDY'S WORD PROCESSOR *

MEETING NOTES JUNE

by Colin Rutherford
Hon. Secretary

Committee. B. Kehlet (President)
C. Rutherford (Sec.)
H. Hoffman (Editor)
G. Saint
M. McGuiness (V.Pres.)
P. Kazacos (Treas.)
D. Riley (Librarian.)
G. Tahmindjis

ATTENDANCE. The general meeting got under way at 6.30 p.m. with the usual roll-up of about 110 members.

APOLOGIES. Advice was received that Bill Hood would not be present and Mark Shane apologised for having to leave as the meeting started.

MINUTES. The previous meeting had been reported in Applecations and the minutes were accepted as read following a motion by E.Mehrtens, seconded by Ben Simmons.

CORRESPONDENCE. Letters from members and potential members were reported by the Secretary; replies had been sent to all. Commercial literature was tabled and passed to the Purchasing officer and Editor. Letters from ACT and Queensland user groups requesting reciprocal membership were received and will be considered by the committee before the next meeting. Receipt of correspondence was acknowledged by a motion from E.Mehrtens, seconded by C. Orton.

FINANCIAL REPORT. The Treasurer advised that we have a current balance of $6203.08 with outstanding payments to be made for printing and a refund to be paid out on a printer order which we could not supply.

→

Moved by E.Mehrtens and seconded by B.Stanley that the Treasurer's report be accepted.

DISK LIBRARY. Two new double sided disks were available this month and old disks 9,10,11 and 12,13 have been consolidated onto two disks. Catalogs were available at the meeting and details would be printed in Applecations. Two disks were received from the International Apple Corps but they were damaged and replacements will be requested.

PURCHASING. Offers are as advertised in the magazine. There was a mistake in the June issue regarding the normal price of Galactic Empire but the price to members was correct. Sandy's FDOS is highly recommended. A new Japanese disk drive was shown and tested at the meeting. It seemed to be compatible in every way with Apple drives and had the same appearance. Brand name and service and warranty details are being sought and a price of well under $700 including tax for one drive with controller card is envisaged. A questionaire is being prepared to include in next months magazine so that members can identify the products that they are interested in.

CONSTITUTION. The amendment to correct some technical errors was passed by a majority vote. Full details were published in the June issue of Applecations and the disk copy of the constitution will be updated as soon as possible.

GENERAL BUSINESS.

NIBBLE subscriptions; Price increases have created a serious situation which will have to be reviewed at the next meeting. See the notice printed elsewhere in this magazine.

INTERNATIONAL APPLE CORP.; Roger Keating, back from the Applefest in Boston gave his impressions of the American scene. Airmail distribution problems are blamed for the latest difficulties in obtaining Apple Orchard magazines. Apple Inc.'s ban on mail order sales of their products is receiving a strong reaction. The IAC is planning to poll member groups to find out how much assistance they receive from the Apple company. Future plans include expanding the Directorate to allow three directors from each continent. Those Ap-Notes that so few of us have seen, are going to be published in professional form and a 450 page volume of Tech. Notes will be distributed to each user group with provision to purchase extra copies. On the hardware side a new Dual disk drive swithchable between 35 and 80 tracks will be available for the cost of two normal drives and yet another Apple look alike, this time from Germany. The IBM personal computer is making big inroads into the sales of other brands. Only Apple seems to be holding it's own. There is not much software for it yet but many programmers are working on that. Roger saw Neil Bennet, an AUG ex-president, who is working on a new version of Hand Holding Basic although not much has been heard of the version that he sold to APPLE.

NEWS AND VIEWS.

-Literature was presented describing an Apple-type computer from Singapore selling for about A$500. Also The Tool, a programme generating utility that seems very good.

-A memory expansion kit is now on sale in the US which allows you to replace the 16K chips with 64K IC's. That's 192K on board and it gets configured as a language card and a phantom disk as well.

-Two second-hand Apple systems were mentioned for sale but they'll have gone by the time you read this.

-Bruce Kehlet announced that he has joined the New Generation company.

→

New products from them will include a card reader for education applications, a PROM burner, and a battery back-up system to beat the power black-outs. They also offer an extensive repair service.

DEMONSTRATIONS.
Chris Orton had managed to get a copy of Apple Speller which he proceeded to demonstrate much to the delight of Michael McGuinness who was prepared to point out alternative spelling for nearly anything.The Speller works on standard text and binary files and is very versatile in correcting and marking mis-spelt words.

A test run of the Japanese disk drive referred to earlier completed the activities for the night.

S C O L A

SCOLA, Sydney Centre of Learning for Adults, is an independent, non-profit organisation which has been formed for the purpose of providing high quality, stimulating adult courses.

These courses are open to everyone and there are no preliminary qualifications, no exams and no certificates. The courses are for people who have not necessarily had any background in the subjects and the point is to learn purely for the sake of interest and enjoyment.

Although some people may be intimidated by the thought of classroom learning, it is important to realise that the courses are informal and students are not pressed to perform in any way. Participation through questions and discussion is welcomed and classes involve stimulating and enjoyable interaction between students and lecturers.

WHERE:
All classes are held at the SCIENCE CENTRE, 35-43 Clarence st. Sydney. This is not far from Wynyard station.

COMPUTING COURSES:

22 INTRODUCTION TO MICROCOMPUTERS

This course is designed for those who have had no previous experience of computers.
The course will include the history of computers as well as the Basics of programming and the practical applications of microcomputers in the home or business.
Apple microcomputers will be available in the class for students to gain some experience through direct use.

Lecturer: Roy Phillips, technical consultant to Computerland Eastern Suburbs.

Mondays 7.45 - 9.45 p.m.
12 meetings, from July 26
Fee $65

23 THE BASIC LANGUAGE FOR MICROCOMPUTERS

A practical and intensive course in the programming of microcomputers. The course is intended for the person who wants an introduction to the BASIC language without previous computer experience. Topics to be covered include elementary operation, BASIC programming, program development, debugging and graphics. The participants will be led through the steps from fundamental operation to the writing of complex progams with the aid of an Apple computer.

Lecturer:Ian Burgess, B.Sc.,B.E.

Mondays 5.45 -7.45 p.m.
10 meetings, from July 26
Fee: $65
OR
Mondays 7.45 - 9.45 p.m.
10 meetings, from July 26
Fee: $65

➡

25 BUSINESS COMPUTING

This course is directed at business managers and professionals at any level who wish to gain a better grasp of the nature of data processing and its effects on business - past, present and future.

The course will include:
1. Historical perspective on data processing.
2. Description of the current scene in business computing.
3. Factors contributing to the success or failure of business system projects.
4. Wider social implications.
5. Horizons.

Lecturer:Peter Boardman B.Comm.

Mondays 5.45 - 7.45 p.m.
12 meetings, from July 26
Fee: $65

ENROLLMENT INFORMATION

To make sure of a place in the class of your choice, it is necessary to enrol in advance. Enrollments will be accepted at the first meeting of a class if there is a vacancy.
Phone 371-8197 during business hours.

EXPENSIVE APPLES

by Ken Ozanne

I have just been reading a 1973 book on data processing, by Elias Awad. The interesting thing was that he listed IBM 360 memory prices, for the (in 1971) brand new 128 X 1 memory chips. Depending on configuration (which we shall ignore) prices ranged from 17 cents to 40 cents per bit!

A minicomputer with comparable processing power to the Apple then cost around $15000 without any peripherals or memory (comparisons must necessarily be quite rough due to changes in technology).

We can now calculate some prices for the 1971 "Apple", excluding peripherals:
** Apple II+ with 16k (includes 16k ROM) - a minimal system.
$59,564 to $119,857
** Apple II+ with 48k - standard system.
$104,128 to $224,715
** Apple II+ with 48k, 16k RAMcard and Integer card (my system).
$140,337 to $309,912
** Apple II+ with ROMcard and 6x128k RAM cards (about as large a system as you can build without an expansion chassis and using only boards currently available in australia).
$1,195,958 to $2,793,726
** Apple II+ with 4 expansion chassis, each fully populated with 1 Megabyte cards (the largest system one could conceivably build with existing components - over $40,000 worth at current prices).
$54,190,404 to $127,506,820

We have been assuming that the special features of the apple would have been available in the 1971 computer. Of course we are aware that some features would have been extra-cost items.

For example, graphics capability of anything like the same standard of our computers would have required a dedicated system worth over $100,000 in itself.

And a synthesizer comparable to the Mountain computer or ALF systems (assuming one were available at any price) would have cost much more than $100,000. Some of the systems then available required dedicated use of a computer like the ibm 360. Even the apple's inbuilt capability would have been very expensive.

I don't think it proves anything, though it might help if your wife demands that you get that worthless object out of the living room.

→

Apple II+ with ROMcard and 6x128k RAM cards (about as large a system as you can build without an expansion chassis and using only boards currently available in australia).
$1,195,958 to $2,793,726

Apple II+ with 4 expansion chassis, each fully populated with 1 Megabyte cards (the largest system one could conceivably build with existing components - over $40,000 worth at current prices).
$54,190,404 to $127,506,820

We have been assuming that the special features of the apple would have been available in the 1971 computer. Of course we are aware that some features would have been extra-cost items.

For example, graphics capability of anything like the same standard of our computers would have required a dedicated system worth over $100,000 in itself.

And a synthesizer comparable to the Mountain computer or ALF systems (assuming one were available at any price) would have cost much more than $100,000. Some of the systems then available required dedicated use of a computer like the ibm 360. Even the apple's inbuilt capability would have been very expensive.

I don't think it proves anything, though it might help if your wife demands that you get that worthless object out of the living room.

---

support our advertisers, they support your magazine

APPLE SYSTEM NOTES

——————————————————————by Ian Webster

The Apple Disk controller patent is one of three patents issued for the Apple computer. The soft approach of this design is the reason for most of the copy protection activity that has been such important part of the development of the Apple. → p10

Further system notes include :

PROTOTYPE CARD BUS SPECIFICATION
The documentation for the Prototype card contains the specifications for the use of the Apple expansion bus.
To follow in Applecations - August.

Dos 3.1 RWTS ROUTINES SOURCE CODE - 14 pages - $ 2.00
Sandy Wiggington's original source of the RWTS routines. The code is similar to the DOS 3.3 RWTS routines.
DOS 3.2 DISASSEMBLY - 32 pages - $3.50
Lee Meador from the Fort Worth AUG was the first person to publish a detailed and systematic dissassembly of DOS. The articles were pulished in the FAUG newsletter.

Due to a limited interest in these fairly lengthy listings, the DOS routines will be available on request only, photocopied, priced to cover costs.

APPLE HELP

Thanks to Washington Apple Pi for some clues.

– Q. I'm working on an inventory program written in Applesoft and am using text files to store data to the disk, but I always get an END OF DATA error when I try to read the data back in. What am I doing wrong?

– A. I have reproduced the relevant parts of the program below.

```
10  D$=CHR$(4)
For writing:
300 PRINT D$;"OPEN DATAFILE"
310 PRINT D$;"WRITE DATAFILE"
320 PRINT A,B,C
330 PRINT D$;"CLOSE DATAFILE"

For reading:
400 PRINT D$;"OPEN DATAFILE"
410 PRINT D$;"READ DATAFILE"
420 INPUT A,B,C
430 PRINT D$;"CLOSE DATAFILE"
```

The problem is that when you input numbers (or strings) from the disk you need some delimiter to indicate where one number ends and the next begins. If you do a PRINT A,B,C to the screen the numbers are separated by spaces, but the Applesoft INPUT command does not recognise the space as a delimiter. Even if spaces were valid delimiters this situation wouldn't work because DOS strips away all the spaces leaving the numbers bunched together when written to the disk. The solution is to explicitly provide a deleimiter between the numbers. One way to do this is to "print" a carriage return after each number.

```
320 PRINT A:PRINT B: PRINT C
```

Each PRINT command results in a number and a carriage return being written to the disk. Another equally valid way of doing this is to print a comma between the numbers.

```
320 PRINT A ;",";  B ;",";  C
```

Or, since semicolons are not needed between variable names and literal strings, this also works.

```
320 PRINT A "," B "," C
```

The easiest way to track down these kinds of problems is to use the DOS command MON I,C,O, so that all data read from, or written to the disk, will be printed on the screen.

– Q. When trying to print out some labels , I found that I couldn't TAB past 40 without starting on a new line.

– A. PRINT TAB(0-40) works well on the text screen and some printer interface cards but on others it goes to the next line. To solve this problem it is better to write directly to $0024 (CH) which stores the horizontal cursor position.

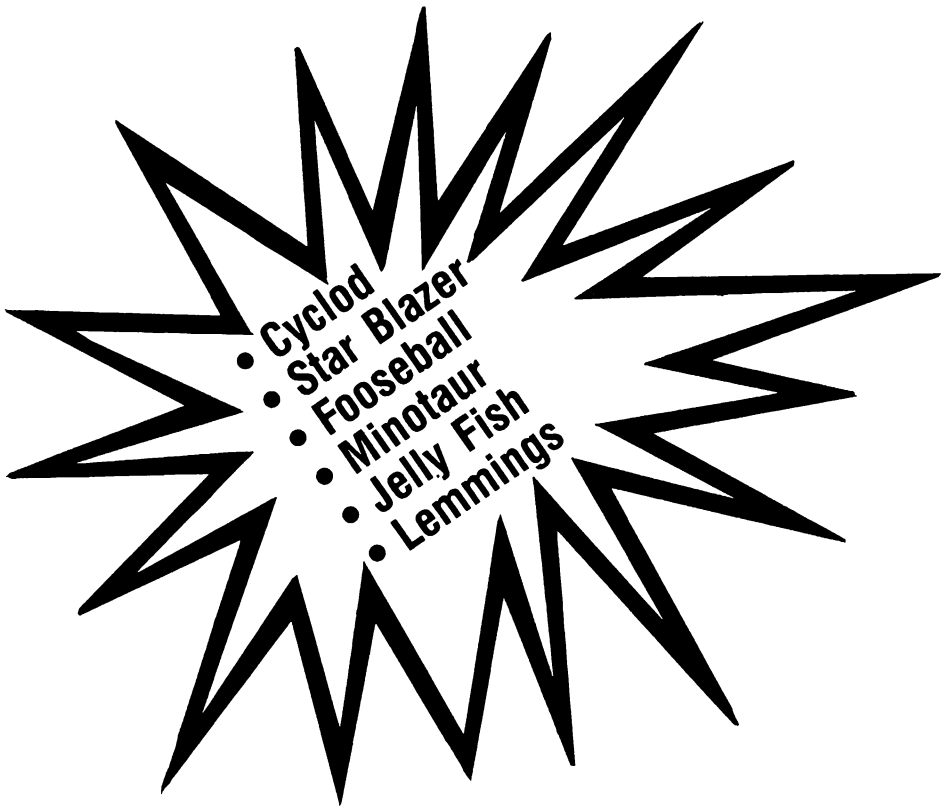
```
1000 POKE 36,I: PRINT  "I  =  TAB
POSN"
```

Interestingly, it is also possible to use this location to read from.

```
1001 POKE 36,(PEEK(36)-1)
```

This will cause the cursor to backspace
The mate of this address is $0025 (CV).This can be used in an interesting technique for checking whether text has reached the bottom of the text screen, you can then stop the display and ask for a reader's response.

```
2000 IF PEEK (37)  <21 THEN  GOTO
4000


4000 PRINT "'RETURN' FOR MORE,
     'ESC' TO MENU"

4010 GET A$: IF A$= CHR$(27) THEN
     1000: REM CHR$(27) = 'ESC'
```



Imagineering offer you the Unique Software package "Our Expertise" in

Business Programmes
Education Programmes
Research Programmes
Communication Programmes
and Fast Action Games

United States Patent [19]
Wozniak

[11] 4,210,959
[45] Jul. 1, 1980

[54] CONTROLLER FOR MAGNETIC DISC, RECORDER, OR THE LIKE

[75] Inventor: Stephen G. Wozniak, San Jose, Calif.

CONTROLLER FOR MAGNETIC DISC, RECORDER, OR THE LIKE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The invention relates to the field of contr llers, particularly controllers for interfacing between a digital computer and a magnetic recorder such as a floppy disc or other memories.

2. Prior Art

Numerous controllers are commercially available for interfacing between digital computers and magnetic disc recorders such as the commonly employed floppy disc recorders. These discs include a plurality of endless, concentric tracks used for storing digital data. The controller typically accepts data in parallel form from the computer and provides the data in serial form to the recorder. Serialized data from the recorder is converted to parallel form for the computer. Controllers perform other functions such as track selection and synchronization.

Commercially available controllers, particularly those for floppy disc recorders, are generally complex and expensive. Because of their cost, they do not lend themselves to the consumer field (e.g., hobby and home uses) or small business use. As will be seen, with the present invention a relatively simple, inexpensive controller is described which is suitable for consumer and small business applications. However, the principles employed in the described controller are applicable to larger, more elaborate systems.

Ofter disc controllers provide track selection signals to the recorder. These signals control a stepping motor which drives the read-write head to the desired track. If the motor is stepped from track-to-track, considerable time is lost in selecting non-adjacent tracks. In the prior art, complex means are used to allow the stepping motor to accelerate and decelerate to and from higher speeds when selecting tracks which are separated from one another by some distance. With the present invention, a computed velocity profile is easily implemented, thus allowing rapid selection of non-adjacent tracks.

When a track is selected, synchronization between the controller/computer and the data recorded on the track is necessary. In some cases, permanent markers, such as holes, are included with each track to provide a fixed reference point. In "soft-sectored" discs, permanent markers are not used. These discs provide wider flexibility since the user is able to format the disc to a particular application. Somewhat intricate hardware is used to provide synchronization with these soft-sectored discs. A method is described in this application which provides rapid synchronization for soft-sectored discs.

BRIEF DESCRIPTION OF THE INVENTION

An interfacing means for interfacing between a digital computer and a magnetic disc recorder, such as a floppy disc or other memories, is described. A serial/-parallel register is employed for communicating with the computer on the data bus. A logic means, which may be a read-only memory, receives input signals (addresses) and provides output signals in response thereto. The logic means is controlled by a timing means which received a synchronization signal from the computer. Output signals from the logic means are coupled to the register and to the timing means. Input signals to the logic means (address signals) are received from the timing means, recorder and resister. Thus, some of the output signals from the logic means are used as address signals for selecting the next output from the logic means.

A method is described for synchronizing an n-bit digital register from the recorder. A synchronization field which comprises a plurality of codes, each having n-bits of one binary state and at least one bit of the other binary state, is coupled to the register. The register is automatically reset each time a bit of the one binary state is moved into the n^{th} stage of the register. With this synchronization field, the register automatically becomes synchronized with the codes and words.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates the controller of the present invention interfacing between a digital computer and a disc driver (recorder).

FIG. 2 is a block diagram of the controller of this present invention.

FIG. 3 is a detailed block diagram of the controller logic and timing means shown in FIG. 2.

FIG. 4 is a graph of the synchronization field used to synchronize a register with recorded data.

FIG. 5 is a graph illustrating the format for each sector of each track, in the presently preferred embodiment.

FIG. 6 is a graph illustrating the byte format employed in the presently preferred embodiment.

DETAILED DESCRIPTION OF THE INVENTION

A controller for providing an interface between a digital computer and a magnetic disc recorder or other memory means is described. While the following description is directed towards a floppy disc recorder, the invented concepts may be employed with other memory means, particularly where data is recorded in serial form such as in a charge-coupled device (CCD), bubble memory, etc. In the following description, numerous specific details are set forth such as specific word lengths, etc., to provide a thorough understanding of the present invention. However, it will be obvious to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known circuits have been shown in block diagram form in order not to obscure the present invention in unnecessary detail.

In the presently preferred embodiment, the described controller and method of synchronization are employed to provide an interface between a microcomputer and a minifloppy disc recorder. The controller is particularly suited for use in the consumer field such as for home, hobby or small business use. In particular, the presently preferred embodiment of the controller is employed to provide an interface between an APPLE-2 computer, manufactured by Apple Computer, Inc. of Cupertino, Calif. and a SHUGART drive, Part No. SA-400, SA-390 or equivalent.

Referring first to FIG. 1, the magnetic disc controller of the present invention, shown as controller 12, interfaces between a digital computer 15 and a disc driver 16. The digital computer 15 is coupled to the controller through a data bus 13 and through an address bus 14. The controller 12 is coupled to the driver 16 through a plurality of control and data lines. The selection of the

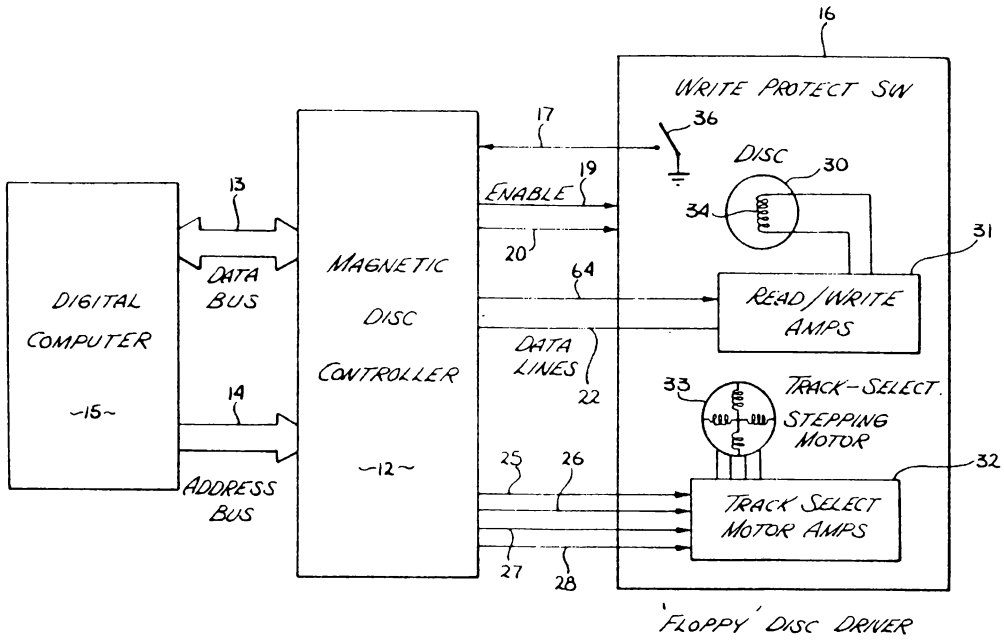


Fig. 1

The Apple Disk controller patent

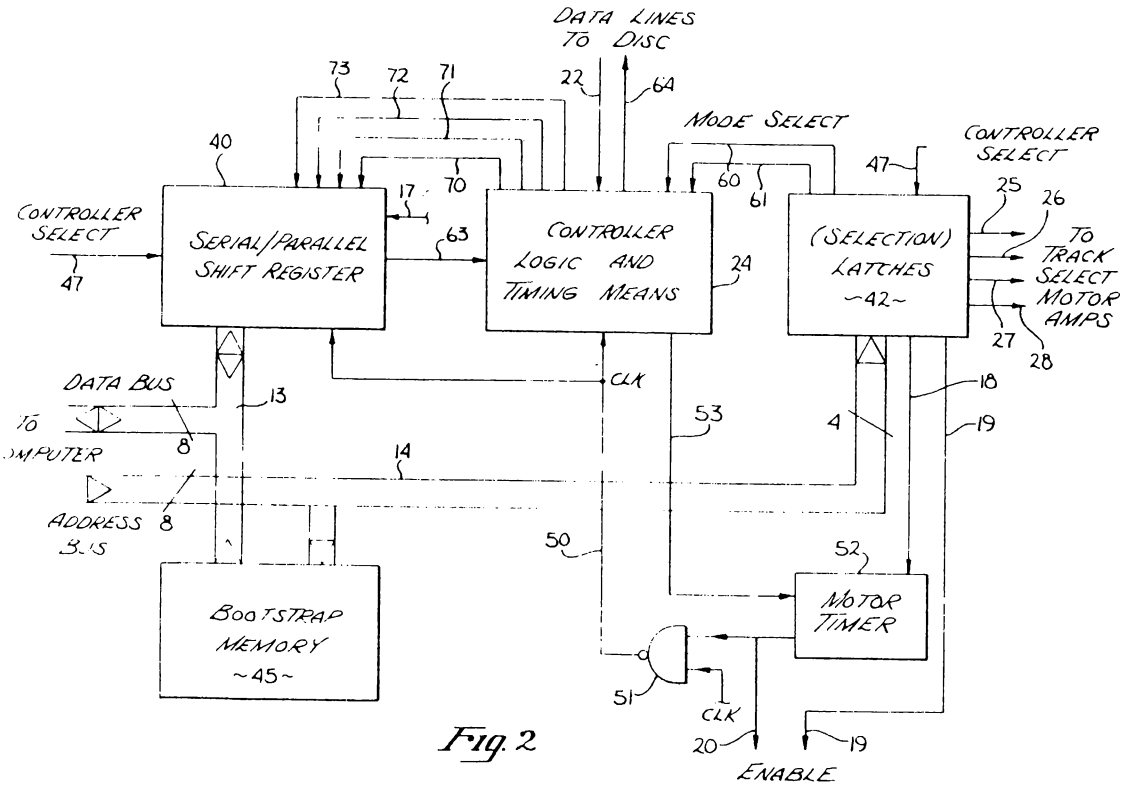


Fig. 2

track is controlled through the signals on lines **25, 26, 27** and **28**, which are coupled to the four phases of the track-select stepping motor **33** through the track-select motor amplifiers **32**. The motor **33** and amplifier **32** are ordinary components commonly employed in disc drivers.

A data signal for recording data onto the disc **30** is coupled to the driver **16** through a line **64** which communicates with the head **34** through the read/write amplifiers **31**. Data read from the disc **30** is coupled through the amplifiers **31** to the controller **12** via line **22**. The head **34** is moved by the stepping motor **33** to the desired track on the disc **30**. Enabling signals to control the driver **16** are coupled to the driver from the controller **12** via lines **19** and **20**. A write protect switch **36** couples a signal **17** to the controller **12** when there is an indication within the driver that the information on a particular disc is not to be erased. This is a common signal employed with numerous disc drivers.

Before describing the controller in detail, the byte format employed in the presently preferred embodiment should be described since it will enable a better understanding of the controller. As shown in FIG. **6**, the byte format consists of 8-bit nibbles. Each nibble consists of four data bits and four clock bits; two nibbles are required to store a byte of data. The clock bits are always binary ones. Thus, two consecutive binary zeros never occur in a normal data field; however, two consecutive binary zeros are employed for markers, as will be described. A nibble does not include either the more significant bits or the least significant bits; rather, the odd data bits, D_1, D_3, D_5 and D_7, are included in one nibble and the even data bits, D_0, D_2, D_4 and D_6, are included in the other nibble. By distributing the data bits in this manner, merging of the two nibbles into a standard byte is much easier. Note that if the nibbles are in parallel registers, a shift in one direction by one stage allows merger of the two nibbles into a single byte.

Referring now to FIG. **2**, the main portions of the controller **12** of FIG. **1** comprise a serial/parallel shift register **40**, a controller logic and timing means **24** and latches **42**. The controller logic and timing means **24** is shown in its presently preferred embodiment in FIG. **3**. The latches **42**, which are ordinary digital latches, store data for selecting modes of operation, tracks on the disc and other control signals, as will be explained in greater detail. Also shown in FIG. **2** are a motor timer means **52** and a bootstrap memory **45**.

The controller of FIG. **2**, which is coupled to the computer via the data bus **13** and the address bus **14**, receives a clocking signal at one input terminal of the NAND gate **51** from the computer. A controller-select signal on line **47** is coupled to the register **40** and latches **42** to indicate that the controller has been selected for operation by the computer. Other well-known control signals and lines such as those associated with power supplies are not shown in FIG. **2**.

The controller of FIG. **2** is coupled to the recorder through the lines **25, 26, 27** and **28**, which are also shown in FIG. **1**. The enable signals on lines **19** and **20**, the data lines **22** and **64**, and the write protect switch signal on line **17** are also shown in FIG. **2**.

The serial/parallel shift register **40** is an ordinary digital register for receiving 8-bit words, in parallel, from bus **13** and from shifting this data, serially, onto line **63** during the writing/reading mode. During the reading mode, data is serially shifted into the register **40** and then removed, in parallel, onto bus **13**. The register **40** is controlled by signals coupled to the register on lines **70, 71, 72** and **73**. These lines originate in the logic and timing means **24**. During the reading mode, as will be described, the data shifted into the register is controlled by signals on these lines. During the reading mode, the register **40** is automatically cleared when its n^{th} stage (8th stage) contains a binary one. Note that with the byte format of FIG. **6**, the first bit of each nibble is always a binary one (clock bit). In the recording mode, data is shifted into the register **40** from right to left. The register **40** is also able to shift data from right to left; this is done, as will be described in greater detail, to sense the signal on line **17**.

While in the presently preferred embodiment an 8-bit register **40** is employed, a 16-bit register or two 8-bit registers may be coupled in series to allow the transfer of 16-bit words to an appropriate bus.

The bootstrap memory **45**, which is coupled to the address bus **14** and the data bus **13**, may be a read-only memory such as a PROM. In the presently preferred embodiment, this optionally employed bootstrap memory is a 256-byte memory used to set initial conditions for operating the system software. The memory may be employed for the reading of operating systems from the disc, or like functions.

The controller logic and timing means **24** is able to sense if the disc is up to speed and provides a signal on line **53** so indicating. This signal through timer **52** and NAND gate **51** prevents the clocking signals from being coupled to line **50** unless the disc is up to speed. The motor timer **52** controls the disc drive motor of the recorder via a signal on line **20**. After data is written or read, the timer **52** prevents the disc motor from stopping for a predetermined period of time (e.g., ten seconds). Note that without this timer a considerable amount of time would be required to wait for the disc to be brought up to speed. The signals on lines **18** and **19**, which are stored within the latches **42**, are used to enable the recorder, including its disc motor.

In the presently preferred embodiment, the latches **42** consist of eight latches which act as a storage means and decoding means. Four lines of the address bus **14** are coupled to the latches **42**. Three of these lines are used to select one of the eight latches and the remaining line is used to furnish data (binary one or zero) to the selected latch. In this manner, 8-bits of data are loaded into the latches **42**. Four of these data bits are used to control the four phases of the track-select motor **33** (FIG. **1**); these bits are coupled to the recorder on lines **25** through **28**. Two of these data bits are coupled to lines **18** and **19** for generation of the enabling signals for the recorder. The remaining two bits are coupled to lines **60** and **61** as will be described in greater detail in conjunction with FIG. **3** to select a mode of operation for the controller.

In typical operation, the computer through the controller of FIG. **2** senses the position of the head (current track) over the disc. Specifically, the track number which is read by the head is coupled to the computer through the register **40**. The computer is able to compute the ideal velocity profile for moving the head to the desired track from the current track with a relatively simple algorithm. Since all four phases of the stepping motor **33** are controlled through the latches **42**, rapid acceleration and deceleration, and higher rates of rotation, are obtainable when compared to stepping the motor from track-to-track.

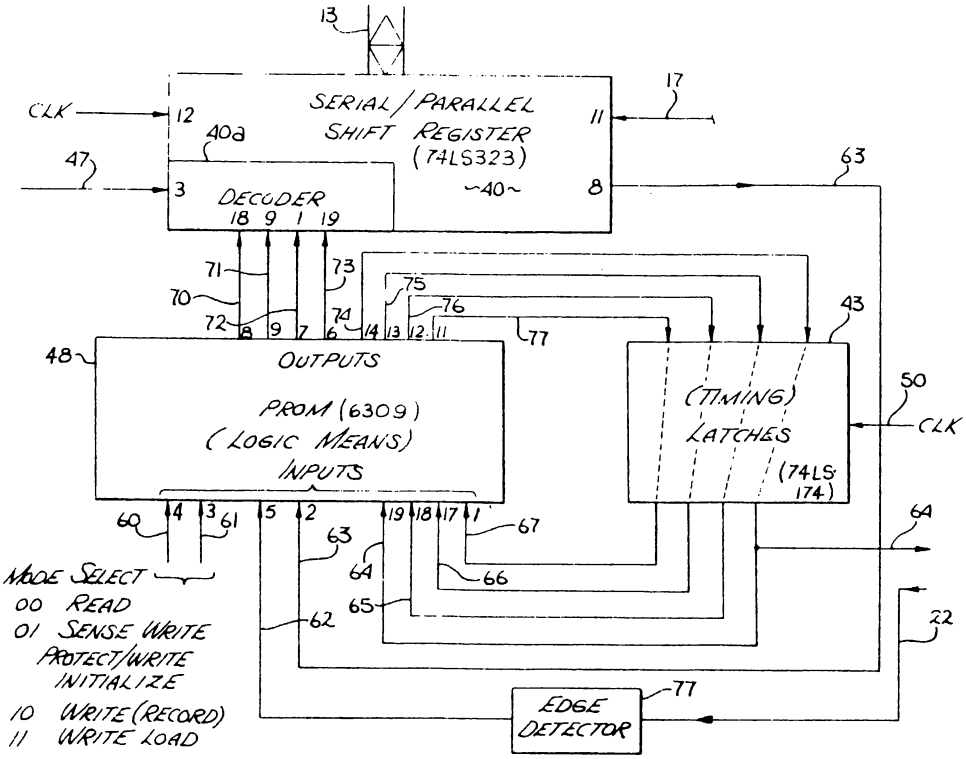


Fig. 3

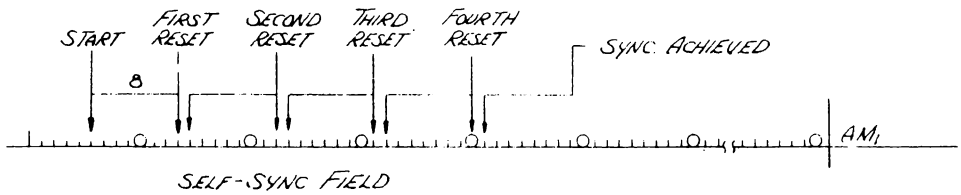


Fig. 4

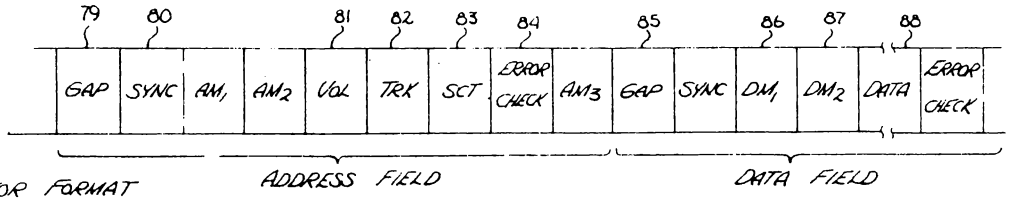


Fig. 5

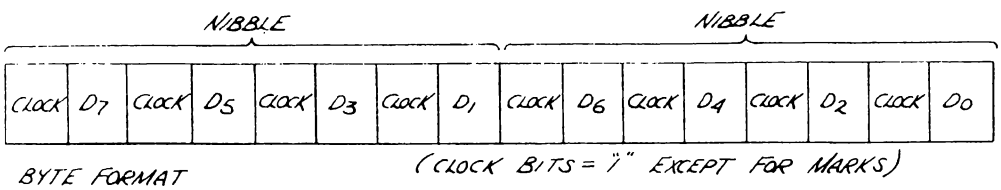


Fig. 6

To achieve an efficient velocity profile for a stepping motor, the prior art often resorted to relatively complex hardware. With the latches **42** and its coupling to the computer through the address bus, an ideal velocity profile may be quickly and efficiently computed without such hardware.

In FIG. **3**, the register **40** and the data bus **13** are again shown. The controller logic and timing means **24** of FIG. **2** comprises a logic means **48** and a timing means **43**. The logic means **48** may be any logic means adaptable for receiving input signals and for providing predetermined output signals in response thereto. Thus, ordinary logic gates or other known logic arrays may be used. In the presently preferred embodiment, a read-only memory, specifically a PROM, is employed. The timing means, in the presently preferred embodiment, comprises four (4) latches which are controlled by the clocking signal, line **50**.

The logic means (PROM) **48**, in the presently preferred embodiment, comprises a 256-byte memory which provides an 8-bit output on lines **70** through **77** for each 8-bit address received on lines **60** through **67**. The specific functions controlled by the PROM shall be discussed below. The specific program stored within the PROM **48** for the currently preferred embodiment is shown in TABLE I.

Two of the address signals to the PROM **48** are coupled from the latches **42** of FIG. **2** on lines **60** and **61**. These signals select the four possible modes of operation; specifically, read (**00**), sense write protect/write initialize (**01**), write (**10**) and write load (**11**). Another input to the PROM **48**, line **62**, is the signal sensed by the recorder head which is coupled to the controller on line **22**. An ordinary edge detector **77** is used for detecting the edge of this signal and for providing a binary signal on line **62**. The input signal on line **63** is the serial output from the register **40**. The remaining four address signals to the PROM **48**, lines **64**, **65**, **66** and **67** are output signals from the latches **43**.

As is apparent from FIG. **3**, four of the eight bits of output from the PROM **48** provide address signals for the PROM. The signals on lines **75** through **77** provide input address signals for lines **63** through **67**. One of these lines, line **64**, also provides the recording signal for recording data onto the disc.

Assume that the controller is in the reading mode as determined by the **00** signal applied to the PROM **48** on lines **60** and **61**. The latches **43** operate at twice the cycle rate of the microprocessor which corresponds to a rate eight times faster than the bit cell disc rate; thus the latches continually release address signals to the PROM. Initially, the register **40** is empty (all zeros). In the presently preferred embodiment, if a transition occurs on line **62** in **11** or fewer of such latch cycles, a binary one is recognized and the PROM provides an output on lines **70**, **71**, **72** and **73** which, after decoding by the decoder **40***a*, shifts a binary one in the first stage of the register **40**. If twelve such cycles occur without a transition, a zero is shifted into the register **40**. (As will be described later, if the first bit is a zero it will be skipped). This continues until the register is full. Counting effectively occurs by the repeated addressing of the PROM as the signals pass through the latches **43**.

The computer senses a full register by polling the data bus and by specifically determining if a binary one is in the n^{th} stage of the register. As mentioned, the first bit of each nibble is always a binary one. When the register is full, the computer removes the data through the data bus **13** and the register **40** is cleared. In the interim, the PROM **48** waits for a binary one and the following bit. Then it writes this binary one and the next binary bit into the register **40**. The temporary delay of shifting into the register is necessary to provide ample time for the computer to withdraw the contents of the register **40**. When a full register occurs, if the first bit sensed by the PROM is a zero, it is effectively skipped, although shifting a zero into the empty register would not affect the operation of the device. In this manner, nibble after nibble is read from the disc, shifted serially into the register **40** and then removed in parallel onto the data bus **13**. The PROM **48** provides the logic to insure the shifting of the correct binary bits into the register **40** as a function of the signal on line **22**, which is coupled to the PROM on line **62**.

In the recording (writing) mode, the mode select signal (**10**) is applied to the PROM **48** on lines **60** and **61**. (Previously, each nibble is shifted into the register **40**, in parallel, from the computer during the write load mode (**11**).) Every eight clock cycles, the signals on lines **70** through **73** cause the register **40** to shift its contents to the right by one stage. For each such shift, the next bit in the register is coupled to line **63**. The signal on line **63** determines the output signal from the PROM, and particularly, the signal on line **74** which is coupled to the recorder via line **64** after passing through latches **43**. Each of the 8-bits are thus shifted from the register **40** and supplied to the recorder.

In the presently preferred embodiment, the mode select signals change to **11** for the loading of the register **40** from the data bus **13**. Note that this is not necessary, and that the computer could directly communicate with the register **40** for purposes of loading data into the register.

During the sense write protect/write initialize mode, the signal on line **17** is shifted to the left by the register **40** and sensed by the computer. In this manner, the computer can determine if the particular disc on the recorder should not be written onto and provide an appropriate indication to the operator. Other data or signals may be transmitted on line **17** where appropriate.

As mentioned, in the presently preferred embodiment, a soft-sectored disc is employed. When the recorder is first selected, signals from the disc are coupled through lines **22** and **62** to the PROM **48** (reading mode). These are shifted into the register **40**. One problem with a soft-sectored disc is that there is no immediate way of determining where in a nibble reading first occurred. Some means or method must be provided to align or synchronize the shifting of the data into the register **40** with the nibbles recorded on the disc.

Referring to FIG. **4**, a self-synchronizing field of coded words are employed to provide synchronization. Each word consists of eight binary ones followed by a binary zero. This self-synchronization field, in a more general form, consists of n-binary ones where n corresponds to the number of stages in the register followed by at least one binary zero. As will be seen with these $n+1$ codes, the register **40** resets with every n bits until the first bit is a binary zero. Then the register resets with every $n+1$ bits.

Referring to FIG. **4**, assume that the disc includes the above-described synchronization field. Assume further that reading begins where indicated by the start line. This first binary one is shifted into the register **40** of FIG. **3**. Eight bits later the first reset occurs and the

register automatically clears since a binary one is in the n^{th} stage. Following this, eight bits later, the second reset occurs. The third reset occurs eight bits later as indicated in FIG. 4, and finally, the fourth reset occurs. (When the fourth reset occurs, a binary zero is in the first stage of the register.) From the next bit forward the register will automatically clear every n+1 bits later, and thus the register will be completely cleared when the address maker (AM_1) reaches the register. Likewise, if reading begins before the synchronization field, synchronization will be achieved before the end of the field.

In the presently preferred embodiment, each of the tracks is divided into 11 sectors; one such sector is shown in FIG. 5. Each sector includes gaps, such as gap 79, to compensate for variations in the disc rate, since the disc is not always driven precisely at the same rate of rotation.

Following the gap 79, there is a synchronization field 80 which corresponds to the field shown in FIG. 4. While in theory only 8 codes of n binary ones and a binary zero are required to synchronize the register, eleven such codes are used within the synchronization field 80 to assure synchronization.

Following the synchronization field, the address marker identified as AM_1 appears on the track. This marker is used to indicate that an address follows. In the presently preferred embodiment, two consecutive address markers are employed as shown by AM_1 and AM_2. The address markers are distinct and immediately recognized by the computer. As previously mentioned with the normal data field and other information on the disc, every other bit is a clock bit (binary one). Thus, two binary zeros do not occur in succession. However, each marker includes both a missing clock bit and data bit. If reading beings in the middle of a data nibble, only a data bit or a clock bit may be missing, but not both. Thus, the computer cannot mistake data or other information for markers.

After the computer senses the address markers, it then knows that address information follows. Three words consisting of a volume number, a track number and a sector number (shown as words 81, 82 and 83, respectively) are read from the discs to provide an address. Following the sector number, an error check is made on the volume, track and sector numbers. In the presently preferred embodiment, an exclusive ORing of these three numbers is employed and checked with the error check word 84. A third address marker, AM_3, is used in the presently preferred embodiment to close the address field.

Next the data field begins, starting with a gap 85. Following the gap 85, synchronization is again required and thus a synchronization field, such as the field 80, is repeated. Two data markers 86 and 87 are used to introduce the stored data 88. In the presently preferred embodiment, 256 8-bit words (256 nibbles) are stored within data 88. Then an error check is made.

No matter where reading begins within the address field or data field of FIG. 5, synchronization is achieved before the computer accepts data. The computer will not accept any data (including addresses) unless it is preceded by a recognized marker. To recognize a marker, the marker must be properly aligned within the stages of the register 40.

By way of example, if reading begins in the middle of the volume number, the data corresponding to this number, the track number, sector number and error check will be serially moved through the register. The marker AM_3 will not be recognized since synchronization has not yet occurred. Following the gap 85 a synchronization field is reached and synchronization occurs. Then the data field markers DM_1 and DM_2 are recognized. However, since they were not preceded by an appropriate address marker, the data which followed is ignored. After a gap (corresponding to gap 79), a synchronization field (corresponding to synchronization field 80) is reached, and synchronization occurs. The markers AM_1 and AM_2 are recognized allowing the identification of the volume, track and sector. Then the data stored within that volume, track and sector will be read, if required.

Thus, a controller for interfacing between a digital computer and a recorder, or the like, has been described. A minimum of hardware is required to fabricate the controller. Synchronization with a soft-sectored disc is achieved without additonal hardware by reading a predetermined self-synchronization field from the disc.

TABLE I

C700-	DA	0D	18	38	0A	OA	0A	0A
C708-	18	39	18	39	18	3B	18	3B
C710-	18	38	08	38	0A	0A	0A	0A
C718-	18	39	18	39	18	3B	18	3B
C720-	OD	OD	28	48	0A	0A	0A	0A
C728-	28	48	28	48	28	48	28	48
C730-	28	48	28	48	0A	0A	0A	0A
C738-	28	48	28	48	28	48	28	48
C740-	0D	0D	58	C8	0A	0A	0A	0A
C748-	58	78	58	78	58	78	58	78
C750-	58	78	58	78	0A	0A	0A	0A
C758-	58	78	58	78	58	78	58	78
C760-	0D	0D	C8	C8	0A	0A	0A	0A
C768-	68	08	68	88	68	08	68	88
C770-	68	88	68	88	OA	OA	OA	OA
C778-	68	08	68	88	68	08	68	88
C780-	0D	CD	C8	C8	0A	0A	0A	0A
C788-	98	B9	98	B9	98	BB	98	BB
C790-	98	BD	98	B8	OA	OA	OA	OA
C798-	98	B9	98	B9	98	BB	98	BB
C7AO-	OD	D9	C8	C8	OA	OA	OA	OA
C7A8-	A8	C8	A8	C8	A8	C8	A8	C8
C7BO-	O9	39	A8	AO	OA	OA	OA	OA
C7B8-	A8	C8	A8	C8	A8	C8	A8	C8
C7CO-	D9	FD	D8	F8	OA	OA	OA	OA
C7C8-	D8	F8	D8	F8	D8	F8	D8	F8
C7DO-	D9	FD	D8	F8	OA	OA	OA	OA
C7D8-	D8	F8	D8	F8	D8	F8	D8	F8
C7EO-	1D	DD	E8	EO	OA	OA	OA	OA
C7E8-	E8	88	E8	O8	E8	88	E8	O8
C7FO-	1D	6D	E8	EO	OA	OA	OA	OA
C7F8-	E8	88	E8	O8	E8	88	E8	O8

I claim:

1. An interfacing means for interfacing between a digital computer and storage device such as a magnetic disc recorder, comprising:

register means having a plurality of parallel input-/output lines for coupling to said computer and a serial input/output line for coupling to said storage device, said register means also having a control means having a plurality of register control lines for controlling transfer of data to and from said register means;

latch means having a plurality of latch input lines and a plurality of latch output lines, said latch means for controlling the flow of digital signals between latch input lines and latch output lines in response to a timing signal,

clock means for generating said timing signal,

read-only memory means having an address means with a plurality of address input lines, and a plurality of data output lines, said read-only memory means for receiving input signals on said address input lines and for providing predetermined output signals on said data output lines in response thereto;

a portion of said read-only memory means data output lines being coupled to said register control means via said register control lines for controlling data flow to and from said register, another portion of said read-only memory means data output lines being coupled to said latch means input lines, said latch means output lines being coupled to said addressing means via a portion of said read-only memory means address lines for providing a portion of an address thereto such that some of said output signals from said read-only memory means are employed as next address signals to said read-only memory means in response to said timing signal;

said interfacing means further including means for coupling the addressing means via another portion of said read-only memory means address lines to said computer whereby said computer may communicate with said storage device through said interfacing means.

2. The interfacing means defined by claim 1 including storage means for receiving digital signals from said computer and for providing control signals to said storage device for controlling track selection, said storage means being coupled to said computer and said storage device.

3. The interfacing means defined by claim 2 wherein said means for coupling the addressing means via a portion of said read-only memory address lines to said computer comprises said storage means.

4. The interfacing means defined by claim 3 wherein said input signal to said read-only memory means from said storage means is used to select a reading mode or a recording mode.

5. The interfacing means defined by claim 4 wherein said storage means comprises digital latches.

6. The interfacing means defined by claim 5 wherein said computer has an address bus and a data bus and wherein said ditigal latches are coupled to said address bus of said computer.

7. The interfacing means defined by claim 1 wherein at least one of said output signals from said read-only memory means is coupled to said storage device to provide a signal for a writing mode.

8. The interfacing means defined by claim 7 wherein said register means serially provides a digital signal on said serial input/output line as one of said address inputs to said read-only memory means during said writing mode.

9. The interfacing means defined by claim 8 wherein said output signals from said read-only memory means controls the serial loading of a digital work into said register means during a reading mode.

10. The interfacing means defined by claim 1 wherein said register means receives digital words in parallel form from said computer and serially couples said words to said read-only memory means during a writing mode.

11. The interfacing means of claim 1 wherein said several input/output line of said register means is coupled to said recorder through said latch means and said read-only memory means.

12. An interfacing means for interfacing between a digital computer and storage device such as a magnetic disc recorder, comprising:

register means having a plurality of parallel input/output lines for coupling to said computer and a serial input/output line for coupling to said storage device, said register means also having a control means having a plurality of register control lines for controlling transfer of data to and from said register means;

latch means having a plurality of latch input lines and a plurality of latch output lines, said latch means for controlling the flow of digital signals between latch input lines and latch output lines in response to a timing signal;

clock means for generating said timing signal;

read-only memory means having an address means with a plurality of address input lines, and a plurality of data output lines, said read-only memory means for receiving input signals on said address input lines and for providing predetermined output signals on said data output lines in response thereto;

a portion of said read-only memory means data output lines being coupled to said register control means via said register control lines for controlling data flow to and from said register, another portion of said read-only memory means data output lines being coupled to said latch means input lines, said latch means output lines being coupled to said addressing means via a portion of said read-only memory means address lines for providing a portion of an address thereto such that some of said output signals from said read-only memory means are employed as next address signals to said read-only memory means in response to said timing signal,

said interfacing means further including means for coupling the addressing means via another portion of said read-only memory means address lines to said computer, said read-only memory means being coupled to receive a data signal from said storage device as an address input on one of said address input lines during reading of data, said read-only memory means being coupled to said serial input/output line of said register means to receive a data signal from said register means as an address input on another of said address input lines, and one of said read-only memory data output lines being coupled to said storage device to provide a data signal thereto during recording of data, whereby said computer may communicate with said storage device through said interfacing means.

13. The interfacing means defined by claim 12 wherein said control signal for said latching means comprises a synchronization signal from said computer.

14. The interfacing means defined by claim 12 including storage means coupled to said computer, said read-only memory means and said storage device for receiving signals from said computer for controlling track selection by said storage device and for selecting said recording or reading of data.

* * * * *

UP THE LADDER GAME

by Stan Gifford.PH 389 7283.

The following game (if you wish to call it that) has an interesting history. Back in the mists of time when I had time to indulge in hobbies (pre APPLE!) I used to do small electronic work. I succesfully built a single board computer (Dream 6800) and a few other goodies. Then for a change I tried to build a Tricky Dicky game which involved pushing a button when a LED was on, and in doing so you switched the next LED up a ladder on, if you hit the button when the LED was off you went down the ladder. I never got the damn thing to work!!! Returning to it one damp day, I decided that I could do a better job via software and the following game ensued.

```
10  TEXT : HOME
20 L1(0) = 200
30 L1(1) = 100
40 L1(2) = 80
50 L1(3) = 60
60 L1(4) = 50
70 L1(5) = 40
80 L1(6) = 30
90 L1(7) = 25
100 L1(8) = 15
110 L1 = 100
120 L2 = 10
130 S2 = 1
140  REM  S2 = LEVEL
150 BUT =  - 16287
160  HOME : PRINT "WHICH PADDLE
    DO YOU WISH TO USE? ";
170  GET A$
180  IF A$ = "0" THEN  GOTO 210
190  IF A$ = "1" THEN BUT =
    BUT + 1: GOTO 210
200  GOTO 160
210  REM
220  REM  L1 IS NUMBER OF FLASHES
230  REM  L2 IS INTERVAL BETWEEN
    FLASHES
240  REM  SC IS CURRENT SCORE
    (HOW HIGH UP)
250  REM  UP THE LADDER GAME???
260  GOSUB 480
270 L2 = L1(SC) / S2
280 L1 = L1(SC) / S2
290 I = 1
300  FOR J = 0 TO L1
310 Z = 0
320  IF I = 1 THEN I = 0: GOSUB
    750: GOTO 340
330 I = 1: GOSUB 750
340  IF Z = 0 GOTO 410
350  IF I = 0 THEN SC = SC - 1
360  IF I = 1 THEN SC = SC + 1
370  IF SC < 0 THEN SC = 8:S2=S2
    - 1: IF S2 < 1 THEN S2=1:SC=1
380  IF SC > 8 THEN SC = 0:S2=S2+1
390 J = 99999
400  GOSUB 480
410  NEXT
420  IF J > 1000 THEN  GOTO 270
430 SC = SC - 1
440  IF SC < 0 THEN SC = 8:S2=S2-1
450  IF S2 < 1 THEN S2 = 1
460  GOSUB 480
470  GOTO 270
480  REM  BACKGROUND ROUTINE
490  GR
500  COLOR= 15
510  HOME
520  HTAB 1: VTAB 23: PRINT
    "LEVEL ";S2;
530  VLIN 0,39 AT 15
540  VLIN 0,39 AT 22
550  HLIN 15,22 AT 0
560  HLIN 15,22 AT 5
570  HLIN 15,22 AT 10
580  HLIN 15,22 AT 15
590  HLIN 15,22 AT 20
600  HLIN 15,22 AT 25
610  HLIN 15,22 AT 30
620  HLIN 15,22 AT 35
630  HLIN 15,22 AT 39
640  ON SC GOTO 660,670,680,690,
    700,710,720,730
650  PLOT 24,39: GOTO 740
660  PLOT 20,38: GOTO 740
670  PLOT 17,34: GOTO 740
680  PLOT 20,29: GOTO 740
690  PLOT 17,24: GOTO 740
700  PLOT 20,19: GOTO 740
710  PLOT 17,14: GOTO 740
720  PLOT 20,9: GOTO 740
730  PLOT 17,4: GOTO 740
740  RETURN
750  COLOR= 15 * I
760  IF  PEEK (BUT) > 127 THEN Z=1
770  IF Z = 1 GOTO 880
780  PLOT 5,5
790  PLOT 5,6
800  PLOT 6,5
810  PLOT 6,6: PLOT 5,7: PLOT 6,7
820  FOR K = 0 TO L2
830 L1% = L1 - J
840  HTAB 20: VTAB 23: PRINT
    "FLASHES LEFT ";L1%;"  ";
850  IF  PEEK (BUT) > 127 THEN Z=1
860  IF Z = 1 THEN K = L2 + 10
870  NEXT
880  RETURN
```

→

Remarks.

I will be honest here, and state that this program was written badly, inasmuch as the logic wanders all over the place. (When I can afford it (for I read 'my wife') I will probably invest in a structured BASIC extension). However I will attempt to explain the program.

LINES 10-100 set up the base timings for each level of the ladder. (No of flashs and interval between flashs).

LINES 110-200 perform other initialization, and sets up paddle to use. (I wish that other games (Nasir please note) would give this option, my paddle zero is well and truly shot playing games).

LINES 480-740 do the background and sets up where blot is up the ladder.

LINES 750-880 is the flash, and detection routine.

LINES 270-470 contain the main logic which does the following:

a) flashes a number of times till a button has been pressed.

b) if a button was not pressed go down a level and go to (a).

c) if button was pressed and flash was out, go down a level, and go to (a).

d) if button was pressed and flash was on, go up a level, decrease no of flashs, increase speed of flashs and go to (a)

SLOT SEARCH

——————————————by Graham Clarke

One would not believe that two Apple II Computers in the same house would cause many problems. Just think, the kids play one and you balance the budget on the other. Or two games of Apple Panic at once. The possibilities are endless.

However, things can go awry when you only have one printer. The answer, simply put a serial card in one and a parallel card in the other. Use the printers ability to switch between the two to access each computer.

Routines were written into business programs to enable printer use but it was impossible to switch the programs between computers. Serial access was programmed differently to parallel.

The Apple Reference Manual hides a wealth of information. One has only to find it and more important understand it. The CSW (Character output SWitch) held the clue. The CSW has two locations $36 (Lo) & $37 (Hi). This pair hold the address of the subroutine which the Apple is currently using. The Monitor's CTRL P command or the BASIC command PR#, can change this address to be the address of a subroutine in a PROM (Programmable Read Only Memory) on a peripheral card. This is the address of the first location in whatever PROM happens to be on the peripheral card.

It was now easy to tell the difference between the serial and parallel cards. The Peripheral Card PROM Locations Table (Page 133) showed $C100 as the first PROM location for Slot 1. It showed 44 ($2C Hex) for the serial and 24 ($18 Hex) for the parallel card. As long as the card has an on-board PROM this location remains constant.

The SLOT SEARCH Routine was written to check all the Slots to see if there were more constant numbers.

```
]LIST

0  REM  ************************
1  REM  *                      *
2  REM  *   > SLOT  SEARCH <   *
3  REM  *                      *
4  REM  *      WRITTEN BY      *
5  REM  *                      *
6  REM  *    GRAHAM CLARKE     *
7  REM  *        SYDNEY        *
8  REM  *                      *
9  REM  ************************
10  HOME : GOSUB 560: GOSUB 690
20  GOSUB 730: HOME
30  DIM A(7,20):G = 1
40  VTAB 4: PRINT "SLOT": FOR X =
     1 TO 20
50  VTAB 4: HTAB 30: PRINT "WAIT
    ";20 - X;: CALL  - 868
60  FOR I = 1 TO 7:N = 49152 + 25
    6 * I:PN =  PEEK (N)
70  VTAB 4 + 2 * I: HTAB 3: PRINT
    I;":PEEK(";N;")=";PN; SPC( 2
    )
80  VTAB 4 + 2 * I: HTAB 21
90 A(I,X) = PN: NEXT I,X
100  FOR A = 1 TO 7: VTAB 4 + 2 *
    A: HTAB 23
110  FOR B = 2 TO 20
120  IF A(A,B) <  > A(A,B - 1) THEN
     PRINT "(EMPTY)": GOTO 500
130  NEXT B:B = B - 1
140  INVERSE
150  IF A(A,B) = 162 THEN  GOSUB
    800: GOTO 500
160  IF A(A,B) = 24 THEN  PRINT "
    PRINTER CARD": GOTO 500
170  IF A(A,B) = 44 THEN  PRINT "
    ASYNC OR SYNC CARD": GOTO 50
    0
180  IF A(A,B) = 8 THEN  PRINT "A
    PPLE CLOCK": GOTO 500
490 E(G) = A: PRINT "<<< WHAT ?":
    G = G + 1
500  NORMAL : NEXT A: PRINT : IF
    G = 1 THEN  END
510  GOSUB 690
520  PRINT : PRINT "AND TELL US W
    HAT YOU HAVE IN:";
530  FOR X = 1 TO G - 1
540  PRINT  TAB( 31)"SLOT "E(X)
550  NEXT X: END
560  PRINT : HTAB 13: INVERSE : PRINT
    " SLOT SEARCH ": NORMAL : PRINT
    : PRINT
570  PRINT "  WE NEED YOUR HELP.
     WE'RE COLLECTING"
580  PRINT "DATA FOR TABULATION.E
    ACH OF YOUR SLOTS"
590  PRINT "1-7, IS REPEATEDLY  P
    EEKED FOR A VALUE"
600  PRINT "WHICH  SEEMS TO BE DE
    TERMINED  BY WHAT"
610  PRINT "PERIPHERAL IS CONNECT
    ED TO IT.  NOTICE"
620  PRINT "THAT  EMPTY  SLOTS
    PRODUCE  VARYING"
630  PRINT "VALUES. PLEASE WRITE
    AND  LET US  KNOW"
640  PRINT "WHAT  VALUES YOU GET
     FOR  YOUR  CARDS"
650  PRINT "IN RETURN, WE WILL SE
    ND YOU A COMPLETE"
660  PRINT "LIST OF ALL TABULATED
      RESULTS. IF YOU"
670  PRINT "WANT, ADD THE APPROPR
    IATE'IF'STATEMENT"
680  PRINT "AFTER LINE 180.": RETURN

690  PRINT : PRINT "WRITE TO:";: HTAB
    (22): PRINT "GRAHAM CLARKE"
700  PRINT  TAB( 17)"89 LAUREL ST
    . WILLOUGHBY"
710  PRINT  TAB( 23)"N.S.W. 2068"

720  RETURN
730  PRINT : HTAB 7: INVERSE : PRINT
    " PRESS ANY KEY TO CONTINUE
    ": NORMAL
740  WAIT  - 16384,128: POKE  - 1
    6368,0: RETURN
800 SLOT = A:D0 = 49280
810 D1 = SL * 16 + D0 + 10:D2 = D
    1 + 1
820 X =  PEEK (D1 - 1):X =  PEEK
    (D1):X =  PEEK (D1 + 2)
830  FOR I = 1 TO 20: IF  PEEK (D
    1 + 2) = X THEN  NEXT I: GOTO
    850
840  PRINT "DRIVE 1";
850 X =  PEEK (D2):X =  PEEK (D1 +
    2)
860  FOR I = 1 TO 20: IF  PEEK (D
    1 + 2) = X THEN  NEXT I: GOTO
    880
870  PRINT " & DRIVE 2"
880 X =  PEEK (D1 - 2): PRINT
890  RETURN
```

```
*********************************
*                               *
* WRONG ADDRESS                 *
* ----------------------------- *
*                               *
*       Do we have your         *
*                               *
*      correct address?         *
*                               *
*           If not              *
*                               *
*  complete the order form      *
*                               *
*         at the back           *
*                               *
*********************************
```

SCREEN SPLIT-2

by Roger Keating

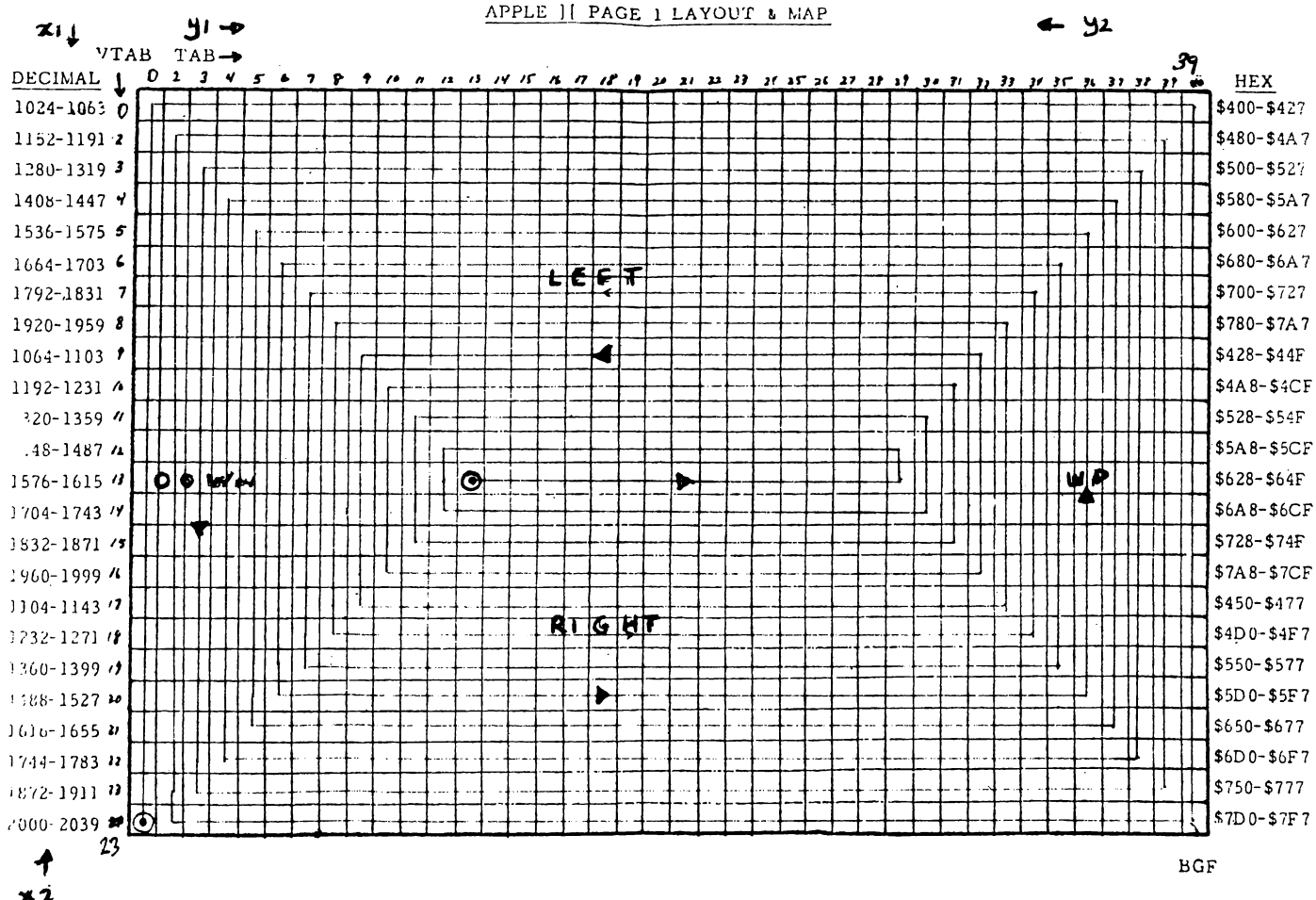
Once again Roger, our War Games expert has provided an interesting screen utility.

The program clears the screen from the centre outward in a snake-like fashion. To clear a coloured screen, change location $339 to $00. The program is not relocatable, and can be used by a CALL 768 from Applesoft.

```
                                ;
INS                             ORG $300
                                OBJ $300
                                ;
THREE SCRATCH VALUES            LINE    EPZ $00
                                CNT     EPZ $01
                                CNT1    EPZ CNT+$1
                                ;
LOW AND HIGH ADDRESS OF THE     L   EPZ $28
LEFT MOST BYTE IN THE LINE      H       EPZ $29
                                ;
                                ;
CALCULATES THE LEFT MOST
BYTE AND STORES RESULT          BCALC   EQU $FBC1
IN $28,$29
                                ;
                                ;
THESE VALUES INDICATE THE       X1      EPZ $03
LIMIT OF THE SCREEN AND ARE     X2      EPZ X1+$1
ADJUSTED AS THE PROGRAM         Y1      EPZ X2+$1
CLEARS THE SCREEN               Y2      EPZ Y1+$1

                                ;
                                ;
SET UP THE COUNTS SO THAT THE PROGRAM REPEATS THE CLEAR
PROCESS 24X40 TIMES SO THAT THE WHOLE SCREEN IS CLEARED
                                ;
                                ;
0300-  A9 18     LDA  #$18  START   LDA #!24
0302-  85 02     STA  $02           STA CNT
0304-  A9 28     LDA  #$28          LDA #!40
0306-  85 01     STA  $01           STA CNT1
                                ;
                                ;
REPEAT THE FOLLOWING 24X40 TIMES
                                ;
                                ;
0308-  A9 00     LDA  #$00  AGAIN   LDA 00
030A-  85 03     STA  $03           STA X1
030C-  85 05     STA  $05           STA Y1
030E-  A9 27     LDA  #$27          LDA #!39
0310-  85 06     STA  $06           STA Y2
0312-  A9 17     LDA  #$17          LDA #!23
0314-  85 04     STA  $04           STA X2
                                ;
FOR THE FIRST SHIFT DOWN AND THEN JUMP INTO THE CYCLE
THAT WILL SHIFT ALL BYTES ALONG THE PATH ONE SPACE
                                ;
0316-  20 4B 03  JSR  $034B         JSR DOWN
0319-  E6 04     INC  $04           INC X2
031B-  4C 25 03  JMP  $0325         JMP START1
                                ;
                                ;
THIS IS THE CYCLE THAT IS REPEATED UNTIL IT REACHES THE CENTRE
OF THE SCREEN (X2=12) AND THE PROCESS IS THEN REPEATED AGAIN
                                ;
031E-  E6 03     INC  $03   START0  INC X1
0320-  20 4B 03  JSR  $034B         JSR DOWN
0323-  C6 06     DEC  $06           DEC Y2
                                ;
0325-  20 95 03  JSR  $0395 START1  JSR LEFT
0328-  C6 04     DEC  $04           DEC X2
032A-  20 70 03  JSR  $0370         JSR UP
032D-  E6 05     INC  $05           INC Y1
032F-  20 A8 03  JSR  $03A8         JSR RIGHT
0332-  A5 04     LDA  $04           LDA X2
0334-  C9 0C     CMP  #$0C          CMP #!12
0336-  D0 E6     BNE  $031E         BNE START0
                                ;
PUT A BLANK IN THE MIDDLE OF THE SCREEN
                                ;
0338-  A9 A0     LDA  #$A0          LDA 0A0
033A-  A4 05     LDY  $05           LDY Y1
033C-  91 28     STA  ($28),Y       STA (L),Y
                                ;
CHECK TO SEE IF THE PROCESS HAS BEEN REPEATED 40X24 TIMES
                                ;
                                ;
033E-  C6 01     DEC  $01           DEC CNT
0340-  D0 C6     BNE  $0308         BNE AGAIN
0342-  A9 28     LDA  #$28          LDA #!40
0344-  85 01     STA  $01           STA CNT
0346-  C6 02     DEC  $02           DEC CNT1
0348-  D0 BE     BNE  $0308         BNE AGAIN
034A-  60        RTS                RTS
                                ;
                                ;
THESE ARE THE SUBROUTINES THAT SHIFT THE LINES ALONG ONE BYTE
                                ;
                                ;
034B-  A5 04     LDA  $04   DOWN    LDA X2
034D-  85 00     STA  $00           STA LINE
                                ;
034F-  C6 00     DEC  $00   DOWN1   DEC LINE
0351-  A5 00     LDA  $00           LDA LINE
0353-  20 C1 FB  JSR  $FBC1         JSR BCALC
0356-  A4 05     LDY  $05           LDY Y1
0358-  B1 28     LDA  ($28),Y       LDA (L),Y
035A-  48        PHA                PHA
035B-  E6 00     INC  $00           INC LINE
035D-  A5 00     LDA  $00           LDA LINE
035F-  20 C1 FB  JSR  $FBC1         JSR BCALC
0362-  A4 05     LDY  $05           LDY Y1
0364-  68        PLA                PLA
```

```
0365-  91 28     STA  ($28),Y       STA (L),Y
0367-  C6 00     DEC  $00           DEC LINE
0369-  A5 00     LDA  $00           LDA LINE
036B-  C5 03     CMP  $03           CMP X1
036D-  D0 E0     BNE  $034F         BNE DOWN1
036F-  60        RTS                RTS
                          ;
0370-  A5 03     LDA  $03    UP     LDA X1
0372-  85 00     STA  $00           STA LINE
                          ;
0374-  E6 00     INC  $00    UP1    INC LINE
0376-  A5 00     LDA  $00           LDA LINE
0378-  20 C1 FB  JSR  $FBC1         JSR BCALC
037B-  A4 06     LDY  $06           LDY Y2
037D-  B1 28     LDA  ($28),Y       LDA (L),Y
037F-  48        PHA                PHA
0380-  C6 00     DEC  $00           DEC LINE
0382-  A5 00     LDA  $00           LDA LINE
0384-  20 C1 FB  JSR  $FBC1         JSR BCALC

0387-  A4 06     LDY  $06           LDY Y2
0389-  68        PLA                PLA
038A-  91 28     STA  ($28),Y       STA (L),Y
038C-  E6 00     INC  $00           INC LINE
038E-  A5 00     LDA  $00           LDA LINE
0390-  C5 04     CMP  $04           CMP X2
0392-  D0 E0     BNE  $0374         BNE UP1
0394-  60        RTS                RTS

                          ;
0395-  A5 03     LDA  $03    LEFT   LDA X1
0397-  20 C1 FB  JSR  $FBC1         JSR BCALC
039A-  A4 05     LDY  $05           LDY Y1
                          ;
039C-  C8        INY         LEFT1  INY
039D-  B1 28     LDA  ($28),Y       LDA (L),Y
039F-  88        DEY                DEY
03A0-  91 28     STA  ($28),Y       STA (L),Y
03A2-  C8        INY                INY
03A3-  C4 06     CPY  $06           CPY Y2
03A5-  D0 F5     BNE  $039C         BNE LEFT1
03A7-  60        RTS                RTS
                          ;
03A8-  A5 04     LDA  $04    RIGHT  LDA X2
03AA-  20 C1 FB  JSR  $FBC1         JSR BCALC
03AD-  A4 06     LDY  $06           LDY Y2
                          ;
03AF-  88        DEY         RIGHT1 DEY
03B0-  B1 28     LDA  ($28),Y       LDA (L),Y
03B2-  C8        INY                INY
03B3-  91 28     STA  ($28),Y       STA (L),Y
03B5-  88        DEY                DEY
03B6-  C4 05     CPY  $05           CPY Y1
03B8-  D0 F5     BNE  $03AF         BNE RIGHT1
03BA-  60        RTS                RTS
```

APPLE][PAGE 1 LAYOUT & MAP



BGF

Program SCREEN SPLIT II Screen

→ Page 37

WHO'S AFRAID OF THE BIG BAD MATRIX

by Keith Brewster

Original article by courtesy Creative Computing December 1981

Beginners in Basic, as well as many people with quite a lot of experience, have real trouble understanding arrays and matrices. This may be because it is a little difficult at first to see the real value of these concepts. Once the techniques are mastered, and it becomes possible to 'visualise' an array or matrix, these extremely powerful features of Basic and many other high level languages become useful tools. The purpose of this aryticle is to introduce these concepts and a few of the techniques involved in matrix/array programming, and to point out some of the places where arrays can be used.

What is an array

An 'array' is like a list. Visualise it as a vertical stack of boxes. The boxes are piled up, and the stack is one box wide and several boxes high. Thus the pile has only one dimension, that of height. It is only one box wide, so it has no width. There is no depth either.

The computer doesn't see things quite this way, but since what it does is totally 'transparent' or 'invisible' to the user, we don't really care. The stack of boxes is convenient for human minds to picture, and we can draw it on a piece of paper. But what good is a pile of boxes if the boxes are empty? Can't we put something useful in them? The boxes are called variables , and they function just as any other Basic variable functions. Each box can store a number or a string. Figure 1 shows how our stack of boxes or variables collectively called an array, should be visualised.

When an array is first DIMensioned (more on this later), it has nothing stored in the boxes.

```
+-----+
!  8  !  (Box 1 contains 8)
+-----+
!  7  !  (Box 2 contains 7)
+-----+
!  3  !  (Box 3 contains 3)
+-----+
!  1  !  (Box 4 contains 1)
+-----+
!  6  !  (Box 5 contains 6)
+-----+
```

--- Figure 1. ---

Most Basics automatically initialise the boxes to 0, but if there is any doubt, it's good practice to initialise them as part of the program. There are many ways to put values in the boxes.

Firstly, they can be 'stuffed' in by direct assignment statements. To do this, we must know the name of the array. Let's call it array A. The boxes have numbers automatically, numbering from A(1) to A(N), where N is the number of the bottom box. In our example, the boxes (called elements) are numbered from 1 to 5. To specify the array and the box to be 'stuffed', we use the following format:

```
ARRAYNAME(ELEMENTNUMBER)
```

For example, in our example, if we want to put 100 into box 3, we can do it like this:

```
A(3)=100
```

In a typical Basic manner, we can use a variable to represent the box, too! Since we are going to get used to using the word 'element' instead of box, we'll choose the variable name E (for element). Now we can do this:

```
E=3: A(E)=100
```

Once gain, the value 100 has been stored in the element that is numbered A(3).

The wheels are going around in the mind, and the question is about to be asked: Why should I use a variable to stand for the element number? Well, you don't need to do this to assign values to elements directly, but what if you want to READ out the DATA statement into your array? Here's how this is done:

```
10 REM STUFF ARRAY A FROM DATA
  STATEMENT
20 DIM A(10)
30 FOR E=1 TO 5
40 READ A(E)
50 NEXT E
60 DATA 8,7,3,1,6
```

This little routine reads the values from your data statement (line 60) and stuffs them in sequence into the array. Note the DIM statement in line 20. This just reserves memory space for a ten-element array. You can DIM an array for practically any number of elements, but remember that unused elements use memory, so don't get carried away! Note, too, that you are allowed a ten-element array without a DIM statement. The word DIM stands for DIMension.

Now we know how to loop using FOR NEXT until our array is full. But what happens if there are elements in the array and only six numbers in the DATA statement? We get an OUT OF DATA error message. What can we do to prevent this? The easiest and most widely used method is top place an end-of-data marker as the last piece of data in the DATA statement. A number - selected so that it would not otherwise appear in the data satement - is used for this purpose. A good example might be 10000. Listing 1 is a typical routine that uses this technique.

```
10 REM EXAMPLE OF END MARKER
20 E=0
30 E=E+1
40 READ A(E)
50 IF A(E)=10000 THEN A(E)=0:
  GOTO 100
60 GOTO30
70 DATA 8,7,3,1,6,10000
100 REM REST OF PROGRAM CONTINUES
110
```

--- Listing 1. ---

Another way to load an array is by use of the INPUT statement. This allows you to put numbers directly into the array from the keyboard. To do this, simply substitute for line 40:

```
40 INPUT "NUMBER, PLEASE ";A(E)
```

When all your numbers are in, you can enter 10000, and the program will jump out of the loop and proceed from line 100.

Before we leave the array and start talking about the matrix, there are a couple more things you should know. In most basics, you can store strings as well as numbers. But you can't put strings in a numeric array. You must DIM the array as follows:

```
   DIM A$(10)
```

Now, anything you put into any of the elements of array A$ (pronounced "A dollar" or "A string") will be stored that way, so you can't put numbers into a string array and expect to do calculations with them! Of course you can use VAL to convert numbers stored as strings to pure numbers.
An exellent example of string use is when you want to sort a bunch of words in (for example) alphabetical order. See listing 2 for an example of this:

```
10 REM *** ALPHABETIC SORT ***
20 REM *** HOUSEKEEPING ***
30 DIM A$(10): REM DIMENSION
  STRING ARRAY TO HOLD UP TO
  10 STRINGS
40 HOME
100 REM *** LOAD STRING ARRAY A$
   FROM CONTENTS OF DATA
   STATEMENT AT LINE 15 ***
110 FOR E=1 TO 6 : REM * NOTE
   THERE ARE 6 ITEMS IN THE DATA
   STATEMENT *
120 READ A$(E): REM * PUT DATA
   ITEMS INTO ARRAY ELEMENS *
130 NEXT E
```

→

```
150 DATA CAN, ARRAYS,NUMBER OR
   WORD,DO,TASKS!,SORTING,
200 REM * SORT ARRAY A$ USING
   ASCII VALUESREPRESENTING
   FIRST LETTER OF EACH ITEM
   IN THE ARRAY *
210 FOR I= 1 TO 6 : FOR E= 1 TO 5
220 IF ASC (A$(E))  >  ASC
   (A$(E+1)) THEN GOSUB 1000 :
   REM * TO SWAP SUBROUTINE *
230 NEXT E
240 NEXT I
300 REM * PRINT OUT SORTED ARRAY
310 HOME
320 FOR E= 1 TO 7
330 PRINT A$(E);" ";
240 NEXT E

999 END
1000 REM *** SWAP CONTENTS OF
    TWO ELEMENTS ***
1010 A$(E)= A$(E+1)
1020 A$(E+1)=TE$
1040 RETURN
1050 END
```

---- listing 2 ----

THE DIM STATEMENT

Now let's take a quick look at the DIM statement. This should go towards the beginning of your program, certainly before the array is used. Remember , too, that more than one DIM for the same array is illegal. Your program will be interupted by a rude error message if you use a GOTO to send the program back to a line before the original DIM statement! So get the DIM out of the way at the beginning of the program, then don't let the program loop back to a line number lower than the DIM line number.
Although Applesoft gives you the first 10 elements without a DIM, get into the habit of DIMming all arrays (and matrices) just in case.
In summary, any time you want to enter a significant number of words, sentences, or numbers using INPUT or want to READ them from disk or DATA statements, use an array.
When you are done print out the list with the program in listing 3

```
200 REM * PRINT CONTENTS OF
   ARRAY A$ *
210 E=0
220 E=E+1
230 IF A(E)=10000 THEN 300
240 PRINT A(E)
250 GOTO 220
300 REM PROGRAM CONTINUES..
```

--- Listing 3. ---

MATRICES

Now bring on those big bad matrices.
No, a matrix isn't something you sleep on (nor is array some kind of disk-shaped fish): A matrix is an array with more than one dimension! Remember we said an array is "one dimensional" because it has only height, and no width or depth? A matrix has more than one dimension, (usually two, but three - and even four dimensional matrices are not really unusual).
To keep it simple we'll discuss two dimensional matrices in this article. Think of a matrix as several arrays, all of the same lenght (height) set side by side. The result is often called a "table", but computer people say matrix. There's one in figure 2.

MATRIX M

	Col.1	Col.2	Col.3	Col.4
Row 1	45	20	65	25
Row 2	30	10	40	20
Row 3	105	200	305	-95
Row 4	300	50	350	250
Row 5	75	30	105	45

--- Figure 2. ---

Anything true of an array is essentially true of a matrix, too. You can assign both Row and Column numbers to variables. The DIM procedure is similar but not identical. The following is a correct DIM statement for our example matrix that has five columns and four rows.:

```
   DIM M(5,4)
```

➡

Notice that the number of rows to be used is always the first number in parentheses, it is followed immediately, without somuch as a space, by a comma, and the immediately by the number of columns you want to DIM. Remember, that the rows are horizontal and the columns are vertical. The DIM statement takes the following form:

```
DIM MATRIXNAME (ROW,COLUMN)
```

For convenience, we'll use these matrix variables: R is the Row variable, and C is the Column variable. Let's assume we want to load the matrix we have created with DIM M (5,4) statemant above, and we want to use the values shown in the eaerlier example. What should the DATA statemant look like, and how can we READ into such a matrix?
The easiest way is two use two nested FOR NEXT loops, one to handle the columns, and the other for the rows. Let's assume that we want to load the matrices horizontally. That is, the first four numbers in the DATA statement will go into row 1, columns 1,2,3,4 in that order. Listing 4 shows how to do this.

```
10 REM STUFF MATRIX FM DATA STMT
20 R=1
30 FOR C=1 TO 4
40 READ M(R,C)
50 NEXT C
60 DATA 45,20,65,25,30,10,40,20,
   105,20,305,-95,300,50,350,50,
   350,250,75,30,105,45
```

--- Listing 4. ---

When we RUN this program the first four numbers in the DATA statement will be stuffed into the first horizontal row of the matrix. Now we need to add an outer loop that will cycle through the Row values, 1 to 5. Listing 5 is the same program with the two loops.

```
10 REM STUFF MATRIX FM DATA STMT
20 FOR R=1 TO 5
30 FOR C=1 TO 4
40 READ M(R,C)
50 NEXT C
60 NEXT R
70 DATA 45,20,65,25,30,10,40,20
80 DATA 105,200,305,-95
90 DATA 300,50,350,250,75,30
100 DATA 105,45
```

--- Listing 5. ---

RUNning this program will load the matrix with the values shown in the example. To print it on the screen, try Listing 6.

```
200 REM PRINT MATRIX CONTENTS
210 FOR R=1 TO 5
220 FOR C=1 TO 4
230 PRINT R;" ";C;" ";"STORES";
   M(R,C)
240 NEXT R
250 NEXT C
```

--- Listing 6. ---

You can "format" the print-out to fit your particular system...
Here are a few ideas for programming using matrices:

* Look-up tables
* Storing numbers or strings for later use.
* Mathematical manipulations such as adding the first column to the second column, and putting the sum in the third column.
* Multiplying a whole potful of numbers by another number.
* Storing files as a table. For example you could write a chequebook balancer that stores old balance in column 1, cheque or deposit in column 2, new balance in column 3, date in column 4, and cheque number in column 5. Or you could store data on people: name in the first column, address in column 2, phone in column 3, etc..

Assuming you have "loaded" a 4 column by 5 row matrix like the one described. here's how to do arithmatic manipulations with it's elements.

```
  MULTIPLY COL 1 BY COL 2

  (Store product in Col 3)

300 FOR R=1 TO 5
310 M(R,3)=M(R,1)*M(R,2)
320 NEXT R
```

→

Notice here that only three Basic lines need to be used to multiply practically any number of elements! Division, addition, subtraction, and the various Basic functions can also be used in this manner. Here's a short routine to look up numbers in a matrix that are related to the number in column 1. For example, if you store the amount of each cheque written in column 1, and you want to print out only those cheques written for a certain amount, you can do it like this.

```
400 INPUT "ENTER AMOUNT "; AM
410 FOR R=1 TO 5
420 IF AM=M(R,1) THEN PRINT
   M(R,1); " ";M(R,2);" ";M(R,3)
   ;" ";M(R,4)
430 NEXT R
```

The program loops through the row numbers, checking column 1 in each row for equality with the number you have entered as AM. When it finds a match, it prints out the contents of the other columns on that row. These columns can be used for practically anything you want! In the case of string matrices, they could be addresses, phone numbers, birthdates, anything you might want to look up. In this case, column 1 would contain the erson's name, and the AM would of course have to be AM$, and the matrix would have to be dimensioned as a string matrix.
By now your curiousity should have been aroused, and you should be thinking of ways to use matrices and arrays in your programming. They are a powerful way to store data. You can READ in the data either from a DATA statement, or from disk. Disk operations are far too system-dependent to cover in a general article, but read your DOS and BASIC operating manuals.
Each system is different in terms of the RAM memory used to store each matrix/array element. This information is in one of the books supplied with your system. If you can't find it, you can write a Basic program that DIMs, then stuffs a matrix by using PRINT FRE(0) that tells your system to print out the remaining user memory. A little experimentation will let you find out how much RAM is used each time you use these powerful features of Basic.
In closing, you should know that matrices and arrays give you one "free" element that you can use any time you DIM an array or matrix! This is row (matrices), or element (arrays) 0. For example, you can store a number like this:

```
A(0)=100
```

Or in a matrix, you can do this:

```
M(0)=100

M(0,1)=101

M(0,2)=102

M(1,0)=103
```

and so forth.
Element 0, Row 0, and Column 0 are always there, and you should be using them if you have very limited memory available.

BUY—SELL—TRADE

FOR SALE:

TELETYPE ASR33 printer/terminal with stand, keyboard, paper tape punch and reader, interface cable. All in good working order,complete with manuals.
Interfaces with the Apple via the SERIAL interface board. Price $200, or $300 with Apple HSS interface board.
Don Riley 451-2475 a.h., 411-7077 bus.

ADVENTURERS' CORNER

with Ed Mehrtens

STRANGE ODYSSEY

Scott Adams of Adventure International has produced a series of 12 adventures , which become more complex and complicated as the series progresses . All of the adventures are well thought out and display Scott Adams well developed sense of humour . Mystery Funhouse , which appeared in this column last month , is also from this series .

This months investigation is the sixth in the series , Strange Odyssey . This adventure is of medium complexity and while not being as easy as Adventureland , No.1 , (recommended for new Adventurers) , it is nowhere near the difficulty of 'Savage Island', No's 10 & 11 . Strange Odyssey is a good adventure with sufficient puzzles for even an experienced Adventurer and will take several sessions to solve . As with all Scott Adams adventures , save the game frequently as your character is bound to be killed while exploring .

The Scenario

You have been forced to crash-land your spaceship on an asteroid due to a power failure , luckily your ship is undamaged but you are trapped far from civilisation . Your object is to return to civilisation , preferably with some profitable items (treasures) .

Some Clues

(1) Can't get out ? Examine the console .
(2) Unbreathable atmosphere ? Wear a space-suit .
(3) Stuck at the top of a cliff ? An asteroid would have low gravity .
(4) Something blocks your path ? Destroy it .
(5) Examine your equipment .
(6) The alien writing can not be deciphered .
(7) A gentle touch works wonders .
(8) The hexagonal room is very important but why is it hexagonal ?
(9) Various combination of action in the hexagonal room produce different results .
(10) Odd occurrences mean something has happened .
(11) High gravity , other than black holes , can be overcome .
(12) Air for your spacesuit is available .
(13) Zoo would contain valuable animals .
(14) All pictures must be capable of being seen .
(15) Collecting treasures is an art .
(16) Ice and snow are similar .
(17) If you need something , look for similar object .
(18) Always examine objects , but I wouldn't twist or push wild animials .
(19) You have to tranquilise some animals .
(20) Any machine should have some means of recomencing a complex cycle .

NOTICE TO ALL ADVENTURERS

Please send details of your adventures to me , to keep this column going .

Help has been requested on the following ;

(1) How do you inflate the boat in Zork .
(2) How do you open the jeweled egg in Zork .
(3) how do you keep the Royal bees alive in Adventureland .

HAPPY MEMBER?

Are you getting value out of your AUG membership - if not - tell us your comments are welcome about Mag.Libr.Meetings.Specials

SOFTSTORY

by John Rotenstein

Review software supplied by IMAGINEERING

NOT SO GREATLY KNOWN SOFTWARE

+ Dos Plus
+ Swashbuckler
+ Audex
+ MasterType

DOS PLUS

Sensible Software

DOS PLUS is the software solution for living with both 13-sector (DOS 3.1, DOS 3.2, DOS 3.2.1) and 16-sector (DOS 3.3) disks.

It adds 8 new commands to DOS. Three of these are built-in and thr remaining five are user definable. The built-in commands allow the user to "flip" between DOS 3.3 and DOS 3.2 with a simple command. The user does not have to re-boot and any programs that reside in memory will not be affected by the flip.

The second command shows the user which DOS is currently in use. The last supplied command shows the user the location of the last loaded Binary file, so that it may be easily saved.

DOS PLUS can be used in your own program, just like a normal DOS command. A ten page booklet accompanies it to explain the use of all the features.

DOS PLUS comes with a DOS command editor. This utility allows you to change any or all of the 28 standard DOS command names to suit your individual preferences. This allows you to use shorter or more familiar names (e.g. "DIR" for "CATALOG"). Modified versions of DOS can be saved by initializing a new disk.

This is a helpful utility if you are constantly changing between different versions of DOS. Unfortunately, it can not be used in conjunction with other DOS versions, such as Sandy's FDOS. Otherwise, it is fine for program use in which extra commands would be helpful.

SWASHBUCKLER

Datamost

Return to those exciting years on the Spanish Main when pirate ships struck fear into the heart of everyone who put to sea!

You're aboard the wickedest pirate ship ever to sail the 7 seas. And you only have the cold steel of your silver sword to protect you! You can charge, retreat, spin around, make your sword swing, slash and stab. But so can your pirate enemies!

All you have is your trusty silver, sword in your fights to the death. You must overcome a horde of pirates, bands of trained killer rats, giant spiders, poisonous snakes and vicious scorpions.

You must fight your way from below the decks of your ship to the top to escape a very large variety of pirates, peg-legs and even samurai. There is amazingly startling HI-RES animation to provide much fun and adventure in every fight, but beware, they will even attack your back.

This is an excellent program with some of the best graphics animation I have yet seen. Certainly worth a good fight.

AUDEX

Sirius Software

Audex is a collection of utility programs which allows you to create sounds, shape them,

→

edit them and play them back in your own Applesoft programs. The only "out of the ordinary" item neccessary is a tape player.

Four programs are included. These are:

* DRAW-A-SOUND is a program for creating and editing sound pulse patterns using the high resolution grahics screen and the keyboard. These "sounds" can be tones, squalks, thumps or anything you choose.

* EXCERPT-A-SOUND is a program for obtaining sounds from the cassette input and preparing part of the audio data for use as sound effects.

* BUILD-A-SOUND is a program to connect sounds and tones together into extended patterns to form music or emulate speech. It can also edit sounds made with the previous two programs.

* AUDIO OPCODES is a collection of machine language routines that allow you to produce sound effects, music, and speech in your own Applesoft programs.

A sixty-five page manual is supplied, which includes tutorials and diagrams on almost every second page. This is a good program to add sound effects to your own program and for experimenting with speech synthesis. Subroutines are unprotected, so they may be used however wished with your own programs.

==========
MASTERTYPE
==========
Lightning Software

MasterType is a computer game designed to teach touch typing. It is a new approach to typing instruction that uses an entertaining shoot-out game to keep players interested and attentive while they learn to type. The program is written in a combination of assembler language and compiled BASIC for maximum speed.

The central character of the game is the MasterType, a wizard who controls a powerful Force. Players tell the wizard where to direct the Force by typing the words which appear on the screen. When the player types one of the words, the wizard appears on the screen and a bolt of lightning flies towards the specific word. This bolt can destroys missiles or other weapons on its path. The object of the game is to destroy the emeny words before they destroy the home base. The players learn to type as they try to win the game.

Although the program is not as instructive as professional courses, it is very good for improving skill and confidence. The HI-RES graphics are fun and provide interest to keep you from becomming bored. A fine program, and a great way to learn typing!

==========
< Pre-Release >
==========

SOFTWARE FLASH-
Escape From Runigstan

+ Bandits + Congo

+ Fly Wars + Cannonball Blitz

** NOTE:
The above five programs are pre-release, and are soon (?) to begin selling.

======================
ESCAPE FROM RUNIGSTAN
======================

A new hi-res adventure from Sirius Software is soon to be released. Very early previews show that the adventure includes b&w drawings (as in hi-res adventure #1: Mystery House), but also has animation spread throughout. ➡

The adventure has ideas never before thought of. While there is animation on the screen, you may type in your next command. Sometimes this is vital, and if you are terribly slow at typing, you may find an early death.

Examples are: catching a mouse before it runs off the screen (although it takes more than that), walking across bridges swaying in the wind, and running and jumping over a gorge. It even seems that there is a skiing section in which you must control your lean.

It appears that this is a program not to be missed, and adventure and graphic fanatics will greatly enjoy themselves.

As to when it shall be released? -- Hopefully soon, so keep an eye out for it!

=========
BANDITS
=========
Sirius software

Your job, simple as it may seem, is to guard the lunar supply base. Unfortunately, every criminal in space is after the supplies and each has a uniquely devious method of trying to get them. Some will come after you with heat-seeking bullets, others that'll drag bouncing balloon-bombs on you.

Armed with a mobile laser gun and protected by a limited amount of shield energy, you score by blasting the creatures into dust. The more of them you blast, the higher you score, but the quicker the Bandits attack.

This game is modelled after the arcade game "STRATAVOX", although those familiar with the origional will miss the voice synthesis which is not present in BADITS. However, another feature adds to the sound. It allows you to not only turn the sound on and off, but loud and soft.

Programmed by the same people who made GAMMA GOBLINS, "BANDITS" is a fine game.

Some of the enemies are very colourful and some are a terrible menice. This program will run on keyboard, joystick/ paddle and, as with most Sirius Software products, on an Atari-type joystick attached to a Sirius joyport.

=======
CONGO
=======
Sentient software

The objective of this game is to navigate a deadly river, rescue survivers from a lost expedition, and find a safe place to dock your handmade raft. You must avoid rocks, unfriendly natives in canoes, and large hippos.

The whole game is in hi-res graphics, created with "The Graphics Magician" by Penguin Software. Although they leave a bit to be desired, it is reasonable for the game.

At particular places throughout the game, there are musical tunes played. This can keep the enthusiasm which may otherwise be lost in this game. Keyboard and joystick is supported.

==========
FLY WARS
==========
Sirius Software

A long, long time ago, in a garage far, far away, a brave alliance of Spider-Fighters were at war with the tyrannical Raygunite forces. You are the last surviving Spider-Fighter. Armed woth a web of purple energy you bravely fight to preserve truth, justice and the Arachnid way of life.

In this game, you control a spider which can leave a purple web behind as a trail. The spider must push one of the many Fly-Fighters on the screen into part of its web, which kills it and places a caterpillar on the screen. You must do the same to the caterpillar, after which a cocoon is placed on the screen.

You then must control your spider to move the cocoon to the top of the screen. For every cocoon pushed to the top, each enemy killed is worth more points.

There are two obstacles which you must avoid. One is a bugspray which sprays a shower of deadly gas which can kill on contact. Another is a beatle which crawls around the screen eating your web and, if possible, YOU!

The game is very enjoyable to play, as each time enough points are gained to clock up another level, not only does it rate you and stir you on (e.g. beginner, spiderman, go to it, attack!), but a random comment is written across the screen by a very clever fly (e.g. spiffy, neato, groovy, superfly).

It would be very hard to become tired of this game, as it gets harder as you play. Included as the cover when you buy "FLY WARS" is a high quality picture which changes depending on the way which it is viewed. This comes with adhesive tape to stick on your favourite wall.

The game can be played with modifiable keyboard controls, Apple compatible paddles or joysticks, or (as with other products of Sirius Software) an Atrai joystick connected to a Sirius Joyport. A truly enjoyable game!

CANNONBALL BLITZ

On-Line Systems

This game, writen by the same person who wrote Pegasus and Jawbreaker, is a modified Apple version of the arcade game "Donkey Kong".

The game has three parts to it. On first screen, you must capture and destroy the emeny's flag. To stop you, the emeny sends down a barrage of cannonballs.

To reach the top level with the flag, you must use levers and balloons which will take you to higher levels.

On the second screen, you must make holes in the platform all the way to the top of the screen and knock the supports out from under the enemy soldier. To make it more difficult, cannons wander around trying to destoy you.

On the third screen, you must once again capture the enemy's flag. There are elevators provided to assist you.

This is a good game, especially for those familiar with the arcade version. Muscial tunes are sounded each time you finsih a new level, and the game can be played with keyboard or joystick. An animated title page also helps you have a (connon)ball of fun.

```
fffff dddd   ooo   ssss
f     d   d o   o s
f     d   d o   o s
fff   d   d o   o  sss
f     d   d o   o     s
f     d   d o   o     s
f     dddd   ooo  ssss
```

IS FANTASTIC!

REVIEWER'S COMMENT

I would like to thank Jodee Rich at IMAGINEERING for supplying new software for review. I have been fortunate to review Pre-Release programs, which have not even been placed on the American software market. This should help you choose your software when it does appear on the market.

John Rotenstein

DISK 1+2+3+4

SIDE 1

*I 056 APPLE ORGAN
*I 026 APPLE VISION
*T 004 CAT
*A 007 MENU
*A 003 INFO
*A 012 PATCHES START
*B 003 PATCHES
*A 020 ELIZA
*T 132 ELDATA
*T 006 ELWORDS
*I 014 ADVANCED SHAPE BUILDER
*B 006 HIRES
*A 023 BLACK BOX
*B 016 BLACK BOX
*A 032 L GAME
*A 055 QUEST
*I 009 IQ TEST
*I 010 LIFE #2001
*I 010 ROUTINES
*I 008 B/BSTAT
*B 009 SINGLE DRIVE COPY
*B 003 FREE SPACE
*B 005 DISK MAP
*A 020 WORD PUZZLE
*B 005 FIX CATALOG

DISK 1+2+3+4

SIDE 2

*A 015 AIRFOIL
*A 064 BIORHYTHM
*I 008 CALC1
*I 002 CALCULATOR START
*T 004 CAT
*I 026 CHASER - GR GAME
*A 003 CONTROL DISPLAY START
*B 002 CONTROL-A$3B0-L$1E
*I 015 ENGINE
*A 004 HEX/DECIMAL CONVERTER
*A 014 HGR DEMO!
*I 065 INFINITE MONKEYS
*A 003 INFO
*A 026 INTEGER 48K
*I 014 LIS'NER PROGRAM
*A 007 MENU
*B 057 NUMBERS
*I 048 PROBABILITY MACHINE
*A 006 ROGER'S APPLE
*I 015 SHOOTOUT
*I 016 TABLES
*B 003 TALK
*I 007 TARGET SHOOT
*I 028 U BOAT
*A 032 USS ENTERPRISE

CP/M DISK 1+2

SIDE 1

044K DIS.ASM
005K DIS.DOC
003K HOW2BS.DOC
006K PTSRCNVT.ASM
003K PTSRCNVT.COM
002K PTSRCNVT.DOC
013K QUOTES.PRN
007K RESOURCE.COM
030K RESOURCE.DOC

CP/M DISK 1+2

SIDE 2

003K -USERGRP.024
004K DUMP.COM
027K DUMP.ASM
004K MAC40.LIB
027K MACRO.LIB
004K OPCODE.LIB
033K RATFOR.COM
003K VOLUME.DOC
016K XDIR.ASM
002K XDIR.COM

DISK 5+6+7+8

SIDE 1

*A 006 AUG DISK # 5+6+7+8
*A 004 BINARY/DECIMAL CONVERTS/7BITS
*T 005 CAT
*A 004 CIRCLE DRAWER
*A 004 CIRCULAR ART
*A 015 CLOCK
*I 036 CODES AND HANGMAN
*A 003 DECIMAL/BINARY 7BITS CONVERTS
*I 015 DECISIONS
*I 003 DISK AIDE
*B 003 DISK AIDE [MACHINE]
*I 093 DISK AIDE [APPLE CORE]
*I 038 DISK AIDE [DOCUMENTATION]
*I 014 HEADING
*I 020 HELLO APPLE
*A 008 IMPOSSIBLE FIGURE
*A 006 INFORMATION
A 013 INSTRUCTIONS FOR STAR LANES
*B 050 INTBASIC
*A 007 LISTER
*A 024 LOGO
*A 007 MENU
*I 058 OREGON TRAIL
*A 012 SPEED READING TRAINER
A 033 STAR LANES
*A 010 TYPE TEACHER

DISK 5+6+7+8

SIDE 2

*A 014 APPLE II+ MINI/ASM
*A 031 APPLESOFT SHAPE MAKER
*A 006 AUG DISK # 5+6+7+8
*I 023 CARWASH
*T 004 CAT
*I 026 DISK HELPER
*I DISK SPEED TEST
*I 004 DISK SPEED TEST INFO
*B 002 DSPEED.OBJ
*A 021 EQUATION OF A LINE
*A 006 INFORMATION
*B 050 INTBASIC
*A 027 INTEGER @ $6000-TAPE
*A 027 INTEGER @ $A000-TAPE
*B 022 INTEGER BASIC-DISK
*A 003 INTEGER LANGUAGES
*A 019 DISK SORT
*A 024 LOGO
*A 021 MASTERMIND
*B 003 MASTERMIND CORE(A$1C00/L$11C)
*A 007 MENU
*I 027 NUMBER THEORY
*B 002 RWTS
*I 039 SPELUNKING
*B 006 SUPPLEMENT
*A 000 TAXMAN
*I 012 TUBS
*I 045 YAHTZEE
*A 002 HELLO
*B 003 INTERGER SUPER LOCK

DISK 9+10+11

SIDE 1

*A 007 AUG DISK # 9+10+11
*A 024 LOGO
*A 011 MENU
*A 006 INFORMATION
*I 020 AIR ATTACK!
*B 080 ALIVADER
*I 018 CARRYING-BALLOON
*I 015 DEATH STAR
*I 033 HI-RES DRAGON MAZE
*A 006 SINGLE KEY MENU
*I 018 SPACE-WAR V
*I 021 SUBMARINE
*A 003 TEXT FILE READER
*I 005 GAMES MENU
*B 022 INTEGER LANGUAGE
*A 002 LANG
*I 012 LUNAR-LANDER
*I 020 NEW FLY KILLER
*I 014 AIR FORCE BOMBER
*A 026 BARN
*I 007 CAR ANIMATION
*A 009 CHECK BOOK BALANCER
*A 006 EQUIPROBABLE
*A 004 HIGHER HIGH-RES
*B 002 L.L.OBJ.
*A 003 LINE LIST
*A 006 POLAR GRAPHICS
*B 034 RADIO
*A 011 RADIOACTIVE
*A 017 SURFACE
*A 008 MENU WRITER
*A 002 ORGAN RUN
*B 002 ORGAN.OBJ
*A 006 ROGER'S APPLE
*B 002 SANDY'S UPDATE
*B 010 UNIV UPDATE
*A 004 UPDATE RUN

DISK 9+10+11

SIDE 2

*A 007 AUG DISK # 9+10+11
*A 024 LOGO
*A 011 MENU
*A 006 INFORMATION
*I 053 ASSM/MON TUTORIAL
*A 054 BASIC TUTOR CODE
*A 002 BASIC TUTOR RUN
*B 011 BASIC.OBJ
*T 022 COMMENTS/WORDBASE
*A 035 WORDBASE
*A 004 WORDBASE RUN
*B 027 DEMUFFIN
*A 002 DEMUFFIN RUN
*B 027 DEMUFFIN16ONLY
*B 028 FID
*A 002 FID RUN
*B 004 FILTER
*A 003 FILTER RUN
*A 026 INTEGER 32K
*A 026 INTEGER 48K
*B 002 PRIN1
*A 003 RELOCATE
*B 003 RELOCATE.OBJ($800)
*B 015 RELOCATE.S(LISA 1.5)
*A 008 CATALOG MANAGEMENT
*A 027 CATALOG MANAGEMENT - EDIT
*T 007 PROG.LIST
*I 051 BLACKJACK

DISK 12+13

SIDE 1

```
*A 007 AUG DISK # 12+13
*A 024 LOGO
*A 011 MENU
*A 006 INFORMATION
*I 015 ADDRESS2
*A 022 ASTRONOMY-EXPOSURES
*B 002 DFIND
*B 002 FIND
*A 003 FIND AND DFIND INSTRUCTIONS
*A 016 GRANDAPPLE
*A 059 GREAT CIRCLE
*A 004 HIGHER HI-RES
*A 005 HIRES TV PATTERN GENERATOR
*A 026 INTEGER 32K
*A 026 INTEGER 48K
*A 006 MEMORY INTERPRETER
*A 008 MENU WRITER
*A 004 POKES
*A 009 RAM TEST 48K
*A 006 ROGER'S APPLE
*A 010 SOFT EGG TIMER
*B 022 INTEGER LANGUAGE
*A 002 SHOOTING ALIEN
*B 080 SHOOTING ALIEN.OBJ
```

DISK 12+13

SIDE 2

```
*A 007 AUG DISK # 12+13
*A 024 LOGO
*A 011 MENU
*A 006 INFORMATION
*A 006 ALIEN HEART
*B 034 BILD1
*T 004 CASH FORMAT
*A 005 CASH FORMAT INSTRUCTIONS
*A 002 CHARACTERS
*A 006 COLLISIONS
*A 002 CONTROL FINDER(1)
*A 003 DA BIN ICH ...
*B 006 DOS3.3LC
*A 013 FFT+
*A 003 GABY
*T 002 HEXEC
*A 011 HULLO
*B 050 INTBASIC
*B 050 INTBASIC.OLDROM
*A 026 INTEGER 32K
*A 026 INTEGER 48K
*A 005 KRISTALLE
*A 009 LACE
*A 004 LANGUAGE CARD
*A 008 LIFE(AS)
*B 004 LIFE.OBJ(AS)
*A 008 MENU WRITER
*B 098 N.BIN
*A 004 NEON
*A 002 ONE LINE
*A 004 ORNATO 2
*A 006 QQQQQ
*A 004 QUARRY
*A 005 RANDOM
*A 006 ROGER'S APPLE
*A 004 SHAKALIN
*A 005 STAR
*A 003 TOKEN LISTER
*A 008 TT
*A 003 UPPER/LOWER CASE SHIFT
*A 005 VASARELY
```

DISK 14

SIDE 1

```
*A 004 APPLEFESTERING SORES
*A 002 AUG DISK 14+
*T 005 CAT
*A 007 CHESS BOARD
*I 017 CONVERT 3.2 -> 3.3
*A 010 CRITICAL PATH ANALYSIS
*A 011 DIFFERENTIAL EQN SOLVER DEMO
*A 006 DL+
*A 020 FORECASTING
*A 013 FUNCTION PLOTTER II
*A 006 FUNCTION ZEROS
*A 004 GAUSSIAN QUADRATURE I
*A 003 GEOMETRIC MEAN
*A 004 GREATEST COMMON DENOMINATOR
*I 033 HAIKU POETRY C.A.I.
*A 016 HANGPERSON
*A 015 HISTOGRAM PLOT
*A 003 INFORMATION
*A 026 INTEGER 48K
*A 016 INTEGRATON
*A 010 KINETIC ART PROGRAM
*A 007 LINEAR PROGRAMMING
*A 024 LOGO
*B 016 MAP
*A 020 MATH WILLIE WORM
*A 007 MENU
*A 008 MENU WRITER
*A 005 MONTE CARLO
*B 027 MUFFIN
*A 004 NORMAL DISTRIBUTION
*I 010 PERPETUAL CALENDAR
*I 023 POKER STUD
*B 020 POKER STUD.X
*A 003 POLAR ANGLES
*A 007 POLAR COORDINATES
*A 006 PRIME FACTORS II
*B 002 RECOVER
*A 006 ROGER'S APPLE
*A 005 ROOTS OF POLYNOMIALS
*A 004 SIMPSON'S RULE
*A 004 SIMULTANEOUS EQUATIONS
*I 005 SPEED READING LETTERS
*A 009 SPEED READING PHRASES
*A 006 SPHERICAL
*A 008 TYPESETTER
*A 015 UNCLE SAM'S JIGSAW
*A 007 VISICALC FILE PRINTER
```

PASCAL DISK 1+3

SIDE 1

```
    AUG:
002 DOZO.DATA
022 DOZO.TEXT
009 DOZO.CODE
002 SWIRL.CODE
006 SWIRL.TEXT
006 DOUBLE.TEXT
003 DOUBLE.CODE
022 SAINTSTUFF.TEXT
008 SAINTSTUFF.CODE
007 DATESTUFF.CODE
022 DATESTUFF.TEXT
006 VIDEO.ASM.TEXT
003 VIDEO.ASM.CODE
006 CHAIN.TEXT
003 CHAIN.CODE
008 STRFUNC.TEXT
003 STRFUNC.CODE
018 NOTES.TEXT
118 <UNUSED>
```

DISK 14

SIDE 2

```
*T 040 APHORISMS
*A 002 AUG DISK 14-
*T 004 CAT
*A 002 EPSON CATALOG DBL STRIKE
*A 002 EPSON GREETING
*A 002 EPSON LOWER CASE
*A 003 EPSON MX80 CATALOG
*A 005 EPSON MX80 DEMO 1
*A 007 EPSON MX80 DEMO 2
*A 024 EPSON MX80 LABEL MAKER
*B 034 EPSON MX80 LABEL.PIC
*A 003 EPSON MX80 LETTERHEAD
*A 015 EPSON MX80 PROGRAM LIST
*A 023 EPSON MX80 REMINDER CALENDAR
*A 033 EPSON MX80 SETUP
*A 010 FOURIER
*A 007 GAUSSIAN QUADRATURE II
*I 004 ID 440 SCREEN DUMP
*B 003 ID 440 SCREEN DUMP.X
*I 003 IDS 255 PRINTER DEMO
*A 003 INFORMATION
*A 007 MENU
*A 005 PAPER TIGER HGR DUMP
*B 003 PAPER TIGER HGR DUMP.X
*B 034 PAPER TIGER.PIC
*A 020 PERIODIC TABLE OF ELEMENTS
*A 030 PRINTER CENTRONICS 779
*I 009 PRINTER IDS
*I 006 PRINTER IDS SETUP 1
*I 010 PRINTER IDS SETUP 2
*A 007 RENAME DISK
*A 011 RENUMBER VOLUME
*B 003 RENUMBER VOLUME.ML
*I 003 SILENTYPE BOLD PRINT
*B 002 SILENTYPE BOLD.X
*A 024 SSM A10 CARD
*I 045 SWORDS & SORCER
*A 009 TEXTREADER(UPPER/LOWER CASE)
*A 006 UNDELETE
*A 007 UNDELETE 16 SECTOR
*A 002 UNDELETE 16 SECTOR ONLY !
*A 008 UNDELETE INSTR
*A 007 UNDELETE INSTRUCTIONS
```

PASCAL DISK 1+3

SIDE 2

```
    ATTACH:
007 SYSTEM.ATTACH
008 ATTACHUD.CODE
026 DOC.0.TEXT
030 DOC.1.TEXT
016 DOC.2.TEXT
022 DOC.3.TEXT
018 DOC.4.TEXT
028 DOC.5.TEXT
024 DOC.6.TEXT
024 DOC.7.TEXT
028 DOC.8.TEXT
002 ADMERG.CODE
006 ADMERG.TEXT
014 LIST.TEXT
004 LIST.CODE
012 FORTFIX.TEXT
004 FORTFIX.CODE
001 NOT.USED
000 <UNUSED>
```

DISK 15

SIDE 1

*A 008 ANIMAL
*A 055 ART AUCTION
*A 002 AUG DISK NO. 15
*A 006 BALL CATCHER
*A 017 BOMBARDMENT
*A 022 BOXED IN
*A 008 DEPTH CHARGE
*A 010 DOGFIGHT
*A 008 DOORS
*A 016 ELEVATE
*A 026 FRENCH MILITARY GAME
*A 020 GOLF I
*A 004 INFORMATION
*A 014 KLINGON CAPTURE
*A 024 LOGO
*A 011 MENU
*A 029 MONSTER CHASE
*A 007 NUMBER GUESS
*A 014 OBSTACLE
*A 008 PETALS ROUND THE ROSE
*A 010 ROTATE
*A 072 STAR TREK
*A 070 STAR TREK SUPER
*A 019 STOCK MARKET GAME
*A 015 WORD PUZZLE

DISK 15

SIDE 2

*A 006 ADVANCE AUSTRALIA FAIR
*A 006 ART ENCORE
*A 002 AUG DISK NO. 15
*A 005 BIRTHDAY DISPLAY
*A 028 BLACKJACK STRATEGY
*I 009 BOWLING
*I 008 BOWLING(MONOCHROME)
*A 012 COMBAT
*A 017 CRAPS BW
*A 010 CREATE A GALAXY
*A 028 FOOTBALL PREDICTIONS
*A 011 FOX AND HOUNDS
*A 003 GOD'S EYE
*A 020 GOLD MINE
*A 022 GOLF II
*A 016 HI Q
*A 003 HIRES SYMMETRY
*A 027 HOCKEY I
*A 021 HORSE RACE III
*A 006 INFORMATION
*A 026 INTEGER 32K
*A 026 INTEGER 48K
*A 008 LITERATURE QUIZ
*A 024 LOGO
*B 006 LORES SLIDE BOWLING
*A 006 LOTTO NUMBERS
*A 026 MAROONED IN SPACE
*A 011 MENU
*A 007 POOLS NUMBERS
*A 003 RANDOM COMPUTER MUSIC
*A 007 RANDOM TUNE GENERATOR
*A 010 ROBOT BW
*A 012 ROLL A DICE
*A 017 SURVIVE
*A 005 TOSS A COIN
*A 006 TWO COINS TOSS
*A 018 TWONKY I
*A 018 WORD MAZE MAKER

DISK 16

SIDE 1

*A 002 AUG DISK # 16
*A 011 MENU
*A 024 LOGO
*A 006 INFORMATION
*A 020 DISKETTE CARE AND HANDLING
*A 003 FREE SECTORS
*A 003 MISPRINTS
*A 012 PROGRAMMING TIPS
*A 005 SHOW A HIRES SLIDE
*B 034 SLIDE COLOUR TEST
*B 034 SLIDE-MONITOR TEST
*A 002 TITLE FRAMER WHITE BARS
*A 002 WHITE TEXT FRAME
*A 004 CIRCLE AREA
*A 004 CIRCLE CIRCUMFERENCE
*A 005 COIN TOSS EXPERIMENT
*A 006 CURRENCY CONVERSIONS
*A 004 SALES TAX CALCULATION
*A 005 STATIONERY SUPPLIES CHECK LIST
*A 013 SCIENTIFIC BOO BOOS
*A 002 SOUND-CRASH
*A 009 SYNONYM
*T 062 BBS
*I 006 MODEM ALARM CLOCK
*A 045 MODEM AUTO DIALER
*A 017 MODEM CHESS
*I 007 MODEM DEMO 1
*I 007 MODEM DEMO 2
*I 006 MODEM DUMB TERMINAL
*A 004 MODEM HELLO
*A 008 MODEM PASSWORD
*A 004 MODEM PHONE PICKUP
*A 013 MODEM PROGRAM EXCHANGE
*I 006 MODEM SELF TEST
*I 007 MODEM SELF TEST PROGRAM
*I 008 MODEM STORE FORWARD
*A 005 MODEM TEXT FILE TRANSFER
*I 002 MODEM.
*B 002 MODEM.ORIGINAL
*B 002 MODEM.X
*A 003 TEXT FILE READ AND PRINT I
*A 007 AUTO NUMBER I
*B 002 AUTO NUMBER I.X
*I 043 BASIC-APPLESOFT
*A 003 BENCHMARK
*A 003 BSTAT I
*T 002 BUTCH
*A 003 BUTCHER
*A 006 INVISIBLE SIGNATURE
*A 003 TAPE SPEED TABLE

PASCAL DISK 2+4

SIDE 1

UTILITY:
018 CATALOG.CODE
008 LISTER.TEXT
003 LISTER.CODE
006 STATS0.TEXT
003 STATS0.CODE
006 SOUNDFX.TEXT
003 SOUNDFX.CODE
024 CATALOG.TEXT
008 SEARCH.TEXT
016 UPDATE.TEXT
179 <UNUSED>

SANDY'S WORDPROCESSOR: will be
---
demonstrated at the July 12 meeting. If you own one, and want to know more about the finer points of this great program, Sandy will be there to help.

DISK 16

SIDE 2

*A 002 AUG DISK # 16
*A 011 MENU
*A 024 LOGO
*A 006 INFORMATION
*A 008 TYPE SETTER
*A 010 CHARACTER DECODER
*A 023 DUMP MEMORY AND ALTER
*A 002 DUMP PAGE
*B 004 DUMP PAGE.X
*A 029 EDIT FILE
*A 023 EDIT FILE DOC
*A 003 FORMAT $
*A 006 FORMAT $ AND CTS
*A 005 FORMAT %
*A 005 IB TO AB
*A 005 LIST STOP AB
*A 008 RESTORE DELETED FILES
*A 005 RESTORE LINE
*A 009 SORT ALPHA
*A 006 SORT ALPHA OR NUMBR I
*A 005 SORT ALPHA SHEL-METZ
*A 007 SORT HEAP
*A 008 SORT HEAP REV
*B 008 SORT ML
*A 003 SORT NUMBERS
*A 005 SORT NUMBR SHEL-METZ
*A 003 SORT SHEL-METZ
*A 007 SORT WORDS I
*A 005 CAPTURE IB AB
*A 016 DISK SUMMARY
*A 030 DISK SUMMARY EDIT
*B 002 DISPLAY ASC PAGE BY PAGE
*A 011 DOG TAG CREDIT LINES
*T 002 DOG-TAG EXEC
*A 014 ERROR HANDLER
*A 005 GREETING SUBROUTINE
*B 003 H14 DRIVER.A$C100.L$FF
*A 004 H14 INSTRUCTIONS
*A 005 POKE BINARY TO BASIC
*A 003 RAM TEST 48K 30 MINUTES
*A 007 SMALL SORT V2
*T 062 TEST DATA 1
*A 005 TEXT FILE CHAR HEX DUMP
*T 016 Z PROG.LIST
*A 004 HELLO
*A 006 COPOGEN
*A 005 H14 DOCUMENTATION
*B 002 H14 DRIVER.$300
*B 002 H14 DRIVER.$390
*B 002 H14 DRIVER.$800
*B 002 H14 DRIVER.$9500
*B 002 H14 DRIVER.APMAIL
*B 002 ASCII
*B 013 ASCII PRINTER DRIVER
*I 005 BAUD RATE ADJUSTMENT
*B 002 DRIVER A768 L176
*A 005 LABEL NUMBERING
*A 002 PICTURE LOADER
*A 011 PRINTER PR40 BANNER

PASCAL DISK 2+4

SIDE 2

PROSE:
032 PROSE.O.TEXT
032 PROSE.B.TEXT
032 PROSE.D.TEXT
032 PROSE.A.TEXT
016 PROSE.E.TEXT
008 PROSE.F.TEXT
034 PROSE.C.TEXT
004 PROSE.TEXT
034 PROSE.CODE
004 CATALOG.4.TEXT
046 <UNUSED>

DISK 17

SIDE 1

*A 007 AUG DISK # 17
*A 024 LOGO
*A 011 MENU
*A 004 INFORMATION
*A 002 APPLE TUTORIAL
*A 025 MENUMENU
*A 017 STRTSTRT
*A 023 HELPHELP
*A 144 PT.1PT.1
*A 086 PT.2PT.2
*A 108 @REF@REF
*I 010 VIDEO TEST
*A 006 DECISION MAKER
*A 006 DICE ROLLING EXPERIMENT
*A 005 EASTER DATES
*A 006 GOING ON HOLIDAYS CHECK LIST
*A 008 LETTER FREQUENCY

DISK 17

SIDE 2

*A 007 AUG DISK # 17
*A 024 LOGO
*A 011 MENU
*A 004 INFORMATION
*A 013 DECISION MAKING AIDE DOC.
*A 007 DECISION MAKING AIDE
*A 025 APPLE SKETCH
*A 008 PRINT CATALOGS
*A 009 SLOT SEARCH
*A 016 APPLESOFT LISTER INSTRUCTIONS
*B 003 APPLESOFT LISTPRINTER
*B 004 PAGE DUMP
*A 023 CALENDAR
*A 005 HEXCON
*I 006 AUTHORSHIP
*I 004 BASIC PROGRAMMING
*I 036 BASIC PROGRAMMING 1
*I 047 BASIC PROGRAMMING 2
*I 048 BASIC PROGRAMMING 3
*I 044 BASIC PROGRAMMING 4
*I 051 MINI ASSEMBLER TUTORIAL
*I 026 TOP DOWN PROGRAMMING
*A 028 DOS SYSTEM INSTRUCTION
*A 014 SUPPLEMENT MINI ASSEMBLER
*B 006 SUPPLEMENT.X
*A 009 WHY COMPUTERS CAN
*A 017 QUESTIONS-COMPUTERS

"APPLECATIONS"

Back Issues available

at the July 12 Meeting

$1.50 each.

DISK 18

SIDE 1

*A 007 AUG DISK # 18
*A 024 LOGO
*A 011 MENU
*A 004 INFORMATION
*I 009 TEXT HELLO
*T 009 TEXT HOW TO
*T 010 TEXT INTRO
*T 006 TEXT PEEK POKE CALL
*T 010 TEXT PROGRAMMING
*T 007 TEXT REDBOOK
*T 011 TEXT SOFTWARE
*I 003 TEXT FILE READ
*I 003 TEXT FILE WRITE
*I 054 PROGRAM DEVELOPMENT PACKAGE
*A 008 BINOMIAL DISTRIBUTION
*A 017 CALCULATOR
*I 010 CALC PI TO 1000 DIGITS
*A 008 EXTERIOR BALLISTICS
*A 011 FAST FOURIER TRANSFORM
*A 012 FFT
*A 013 FFT+
*I 008 HEX-DEC CONVERTER
*A 015 HARMONIC ANALYSIS
*A 012 HEX CONV
*A 006 LOOP ANTENNA
*I 008 LONG DIVISION
*A 015 METRICS LENGTH
*A 026 METRICS VOLUME
*A 017 METRICS KITCHEN
*A 020 METRICS TEMPERATURE WEIGHT
*I 009 MULTIPLY
*I 013 MATH PRACTICE
*I 009 NUMBER BASE CONVERTER
*A 007 PERMUTATIONS-COMBINATIONS
*A 006 PRIME FACTORS OF INTEGERS
*A 006 PLOT
*A 006 QUADRATIC SURFACE
*A 010 REACTANCE CALCULATIONS
*A 005 SIN PLOT
*A 005 TRANSISTOR PARAMETERS
*A 006 XLINE IMPEDANCE
*A 009 APSOFT MUSIC TONE TESTER
*A 003 APSOFT SOUND EFFECT TESTER
*A 010 GUITAR PITCH PIPE
*A 004 INTERESTING SOUND EFFECTS
*A 008 JUST BONES
*A 006 DRAMA FESTIVAL CHECK LIST

APPLEFEST 1982

SIDE 1

*A 025 HELLO
*A 007 MENU
*T 003 CAT
*A 005 TEXT READER
*T 020 DISK PROGRAM COMMENTS
*B 022 INTEGER LANGUAGE
*A 002 LANG
*I 005 GAMES MENU
*B 034 TOLLSCENE
*I 012 LUNAR-LANDER
*I 014 AIR FORCE BOMBER
*A 019 TOLLBOOTH - THE BITS WINNER
*I 020 NEW FLY KILLER
*B 080 ASTEROYDER
*I 051 BLACKJACK
*B 080 SHOOTING ALIEN
*A 006 ALIEN HEART
*A 002 ONE LINE
*B 006 RATS! BINARY
*B 002 TOLLSHAPES
*I 025 RATS! A 3-D MAZE PERSPECTIVE
*A 002 RUN ASTEROYDER
*A 002 RUN SHOOTING ALIEN
*A 012 WORMS
*A 022 AWARI

DISK 18

SIDE 2

*A 007 AUG DISK # 18
*A 024 LOGO
*A 011 MENU
*A 004 INFORMATION
*I 055 CONCENTRATION
*I 017 CONNECTION
*I 062 CRIBBAGE
*I 047 HOOVER DAM
*A 029 STOCK TRADER
*I 016 LIFE
*A 008 CHICKEN
*A 005 APPLE ROSE
*A 017 ZOMBIE ISLAND
*A 009 FIVE GUESSES
*A 009 NIGHTMARE
*A 007 KEYNOTE
*I 008 ORGAN
*A 020 INVADERS
*I 024 INVISIBLE TANK
*A 005 SPIRO
*A 004 MULTICOLOR 1
*I 002 ROAD RACER INTRO
*I 007 ROAD RACER
*B 026 RACER MACHINE
*I 011 STARWARS SOUND EFFECTS
*I 019 COMPUTER MUSIC
*A 008 AWARENESS TEST
*I 034 BRAIN BUSTERS

APPLEFEST 1982

SIDE 2

*A 007 HELLO
*A 062 FILE CABINET 2.1
*A 008 MENU WRITER
*B 007 B.MAS.CAT.48K
*A 018 MAS.CAT.48K.REVISED
*T 002 CASH FLOW FILE
*A 055 CASH FLOW MANAGEMENT
*A 031 DOS FILE TUTORIAL
*T 007 JONFILE
*T 008 MEMMAP48K
*A 005 MEMORY MAP INSTR.
*A 009 SOUND DEMO
*T 006 ADD AMPERSORT TO FC 2.1
*B 003 B.GARBAGE (ORG $9400)
*B 002 B.SCREEN.EDIT.SUBS
*A 008 CATALOG MANAGEMENT
*A 027 CATALOG MANAGEMENT - EDIT
*A 008 EXTRA SPACE
*T 005 CAT
*A 027 FORTRAN INTERPRETER
*A 011 GARBAGE HELP
*A 005 KLEM'S DISASSEMBLER
*A 006 POORBOY'S PLE
*B 002 POORBOY'S PLE.
*B 003 PRETTYLIST
*A 004 PRETTYLIST INFO
*A 017 S-C SCREEN EDIT
*T 025 S.GARBAGE COLLECTOR
*T 010 S.SCREEN.EDIT.SUBS
*A 002 RUN B.GARBAGE (ORG $9400)
*A 003 TEST GARBAGE COLLECTOR
*A 045 CREATIVITY TEST
*A 025 JON 1.0- A PROGRAM THAT LEARNS

BULK PURCHASING SPECIALS FOR JULY

By Ed Mehrtens , Bulk Purchasing Officer .

NIBBLE MAGAZINES

First the bad news, as you may know, airmail rates have more than doubled and we are now being asked to pay the extra cost.
The price we charged our members obviously did not include provision for this increase, therefore we will need to re-negotiate the deal. Would the 30 subscibers let the committee know which of the three options they prefer.

1 Pay the extra (comes to over $7 an issue).
2 Use sea mail (cheaper but a 6 week delay) .
3 Cancel and take out a private airmail subscription.

Please let the committee know which option you prefer, otherwise the decision may not be what you wanted.

THIS MONTHS SPECIALS

This month there are two games plus a disk drive with controller . All are excellent value .

*** DISK DRIVE PLUS CONTROLLER

This disk was demonstrated at our last meeting and showed that it could handle all protected disks, half tracks and nibble counts included. Please note, this is an 'Apple look-alike' and is not an Apple disk drive. The unit is of Japanese manufacture and the circuitry is obviously different while performing the same functions. In fact the only difference I could find was that the Dealer Test Disk could not test the drive circuitry. The speed test worked however and the contoller card handled both the new drive and the Apple disk drives. At $625 however (including controller) this is a real bargain, but I need to order 5 units to get this price. My contact will enquire about the price minus controller .

*** SANDY'S WORDPROCESSOR

An excellent piece of software for those of us who write letters, articles, you name it. It probably the best word processor around. As well it has excellent Mailing letter, with personalised inserts, as well as very good Labeller. See the review in 'Applecations', May 1982. If you are in doubt about this program, see it demonstrated next meeting, or speak to our Magazine Editor, who uses it to write the magazine.
Very good value at $160, normally $195.

GAMES

This month we have two different but exceptional games, Wizardry and Swashbuckler.

*** WIZARDRY

This is the best 'Dungeons and Dragons' style game available, it has had rave reviews in the USA, I have it and really enjoy it. I have not reviewed it in Adventurers Corner because I have not completed it. However this does not mean leaving a game half played, rather it is a series of playing exiting and returning as you build your characters. Characters can be of 5 different races, 8 classes, 50 spells, hundreds of magical items and hundreds of monsters. The game is extremely well written and while easy to play is complex by virtue of its immense size, and new scenarios will become available (No.2 has been released in the USA). If you are at all interested in 'Dungeons and Dragons' style games then Wizardry is the only game to consider . To paraphrase Dr. Johnson ' the man , sir , who is bored with Wizardry , is bored with life !' .
Yours for $45 , normal price is $50 save $5 .

→

*** SWASHBUCKLER

An arcade style game where fast reflexes are essential . You are alone , aboard a pirate ship which has the most evil , back-stabbing , crew of cut-throats as ever hoisted the skull and bones . Not only do you have to deal with hundreds of pirates (only two at a time) but also snakes , scorpions , rats and spiders . The pirates are armed with a variety of weapons including spiked clubs , hatchets , cut-lasses , swords , pikes , boat hooks and one is a Japanese Samurai . There are various levels and if the same pirate appears on a higher level , he will be faster and more deadly . None of the pirates have even heard of 'fair play' and will gleefully stab you in the back , beware . You start with three lives and get another after killing 25 pirates . This is an excellent game and is yours for $28 .

NEXT MONTH:

*** VISION-80, Zofarry Enterprises will be making available this top-selling 80-column card at a special price to club members. See next 'Applecations' for details.

SUMMARY:

(1) DISK DRIVE c/w controller $623

(2) SANDY'S WORDPROCESSOR $160
normally $195, save $35

(3) WIZARDRY $45
normally $50, save $5

(4) SWASHBUCKLER $28
normally $ 38, save $10

Order your specials using the order form on the last page, enclosing the correct money, and your Applecations mailing sticker as proof of membership.

QUESTIONAIRE

A questionaire has been included in this months magazine , so that I know which products are of greatest interest to you. Please note , this is not a purchase order and does not even have your name on it . From your replies to the various products , I will be able to select those items most wanted by the members and contact the various firms to get the best prices . Please fill in the form and return it to me , otherwise I must assume you are not interested in any items .

→ Screen Split-2

```
0300- A9 18 85 02 A9 28 85 01
0308- A9 00 85 03 85 05 A9 27
0310- 85 06 A9 17 85 04 20 4B
0318- 03 E6 04 4C 25 03 E6 03
0320- 20 4B 03 C6 06 20 95 03
0328- C6 04 20 70 03 E6 05 20
0330- A8 03 A5 04 C9 0C D0 E6
0338- A9 A0 A4 05 91 28 C6 01
0340- D0 C6 A9 28 85 01 C6 02
0348- D0 BE 60 A5 04 85 00 C6
0350- 00 A5 00 20 C1 FB A4 05
0358- B1 28 48 E6 00 A5 00 20
0360- C1 FB A4 05 68 91 28 C6
0368- 00 A5 00 C5 03 D0 E0 60
0370- A5 03 85 00 E6 00 A5 00
0378- 20 C1 FB A4 06 B1 28 48
0380- C6 00 A5 00 20 C1 FB A4
0388- 06 68 91 28 E6 00 A5 00
0390- C5 04 D0 E0 60 A5 03 20
0398- C1 FB A4 05 C8 B1 28 88
03A0- 91 28 C8 C4 06 D0 F5 60
03A8- A5 04 20 C1 FB A4 06 88
03B0- B1 28 C8 91 28 88 C4 05
03B8- D0 F5 60 FF FF FF FF FF
```

BUY—SELL—TRADE

FOR SALE: - Books

* "Apple II Word Processing" - Evaluations - $10 - cost $20
* "SKARBEKS Software Directory" - $7
* "More Basic Computer Games" - $6 - cost $12
* "Small Business Programs" - University Software - $25 cost $50
* "Home & Economic Programs" - University Software - $15 cost $28
* "Your Computing Magazine" - Issues 2-9, $10

PROGRAM:

* "Locksmith 4.1" with documentation $75 - cost $100

R.Hodkinson, P.O.Box 550,
WAGGA, 2650
Telephone: (069) 22-5451

Apple Users Group (Sydney)

ORDER FORM

Name:.....Phone:.....

Address:.....

Suburb:.....Address label enclosed?.....

State:.....Post Code:.....

New Membership :
Joining Fee $15 + Annual Sub (2nd. half year) $7.50.....= $.....

SOFTWARE LIBRARY:

AUG 1+2+3+4 / AUG 5+6+7+8 / AUG 9+10+11 / AUG 12+13
AUG 14 / AUG 15 / AUG 16 / AUG 17 / AUG 18 / A-FEST'82
PASCAL 1+3 / PASCAL 2+4 / CP/M 1+2

Please send me the following disks :
Number:

( ) x ( ) ..... @ $8 = $.....
( ) x ( ) ..... @ $8 = $.....
( ) x ( ) ..... @ $8 = $.....
( ) x ( ) ..... @ $8 = $.....
( ) x ( ) ..... @ $8 = $.....

Mailing cost $1 per 3 disks.....(x) = $.....

Bulk/Purchase Offers:June-July

SANDY'S FDOS.....- @ $21.... - (x) = $.....
GALACTIC EMPIRE.....- @ $19.... - (x) = $.....
TAWALA'S LAST REDOUBT.....- @ $23.... - (x) = $.....

Bulk/Purchase Offers:July-Aug

DISK DRIVE c/w controller .....- @ $623... - (x) = $.....
** Cartage extra as required **
SANDY'S WORD PROCESSOR.....- @ [redacted]...- (X) = $.....
WIZARDRY.....- @ $45.....- (x) = $.....
SWASHBUCKLER.....- @ $28.....- (x) = $.....
Mailing cost $1 per disk..... - (x) = $.....

Please don't mail, I will collect at meeting ( ) ========
Cheque/Money order for TOTAL..... :$.....

N.B.: All mail is surface, Air Mail is extra.

Please send the order and cheque to:

APPLE USERS GROUP (Sydney) or : AUG Mail Box,
P.O.Box 505 Computerland,
BANKSTOWN, N.S.W.,2200. 31 Market st, Sydney.

Stick address label here

The group meets at 6.30 p.m. on the second monday of each month (tuesday after a holiday monday) at SYDNEY GRAMMAR SCHOOL,Stanley st. SYDNEY.

We share experiences, demonstrations, trade ideas, goods, and discuss developements in the Apple world.
Special interest groups cover Pascal, C/PM, and hardware.

Benefits of membership are:

Regular issues of 'APPLECATIONS'
Access to our Software library
Low-price Bulk/purchase goods.

To become a member, please complete the Order form, enclose the correct money, and send it to the AUG at either address.

Company, Institution, or associate Club membership is available.

APPLECATIONS

Advertising rates:-

i/s front/back $40

Full page $30

Half page $20

Copy must be black on white paper

Advertising may be arranged by phoning Bruce Stanley at (02) 98-6972 a.h.

Stores wishing to obtain copies of the magazine for counter sales should contact the editor.

Copy is always needed for the magazine. This can programs, or useful subroutines, with suitable documentation. The greatest value is obtained when a technique can be understood by the readers so that it can be further utilised.
Reviews are welcome because they enable other Apple owners to assess the usefulness of a product before buying.
Hardware modifications should be supplied with clear constructional details.

All material is preferred supplied in text file format on a disk, which will be returned after copying.